Sonophone: Desarrollo y evaluación de un sonómetro profesional para iOS

Proyecto Fin de Carrera

Diego Torres Domínguez

Escuela Universitaria de Ingeniería Técnica de Telecomunicación
Universidad Politécnica de Madrid

E.U.I.T. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA:

TÍTULO: SonoPhone - Desarrollo y Evaluación de un sonómetro profesional para iOS

AUTOR: Diego Torres Domínguez

TUTOR: Lino Pedro García Morales **Vº Bº.**

DEPARTAMENTO: DIAC

Miembros del Tribunal Calificador:

PRESIDENTE: CARMEN COUSIDO MARTÍNEZ-CONDE

VOCAL:

VOCAL SECRETARIO: ANTONIO MÍNGUEZ OLIVARES

DIRECTOR:

Fecha de lectura: 10/9/2013

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

La medida de la presión sonora es un proceso de extrema importancia para la ingeniería acústica.
Por otra parte, la ubicuidad de los dispositivos móviles inteligentes (smartphones, tabletas, etc.), dispositivos que integran potencia de procesado, conectividad, interactividad y una interfaz intuitiva en un tamaño reducido, abre la posibilidad de su uso como sistemas de medida.
En este proyecto se pretenden utilizar las capacidades de entrada y salida, procesado, conectividad inalámbrica y geolocalización de los dispositivos móviles basados en iOS (iPhone, iPad, iPod touch) para implementar un sistema de medidas acústicas que iguale o supere las prestaciones de los sonómetros existentes en el mercado.
El Sonophone permitirá, mediante la conexión de un micrófono de medida adecuado, la realización de medidas de acuerdo a las normas técnicas en vigor, así como la posibilidad de programar, configurar y almacenar o transmitir las medidas realizadas, que además estarán geolocalizadas con el GPS integrado en el dispositivo móvil. También se permitirá enviar los datos de la medida a un almacenamiento remoto en la nube.
Además de implementar la aplicación, se ha realizado una prueba de funcionamiento para determinar si el hardware del iPhone es adecuado para la medida de la presión acústica de acuerdo a las normas internacionales.

La medida de la presión sonora es un proceso de extrema importancia para la ingeniería acústica, de aplicación en numerosas áreas de esta disciplina, como la acústica arquitectónica o el control de ruido. Sobre todo en esta última, es necesario poder efectuar medidas precisas en condiciones muy diversas.

Por otra parte, la ubicuidad de los dispositivos móviles inteligentes (*smartphones*, tabletas, etc.), dispositivos que integran potencia de procesado, conectividad, interactividad y una interfaz intuitiva en un tamaño reducido, abre la posibilidad de su uso como sistemas de medida de calidad y de coste bajo.

En este Proyecto se pretende utilizar las capacidades de entrada y salida, procesado, conectividad inalámbrica y geolocalización de los dispositivos móviles basados en iOS, en concreto el iPhone, para implementar un sistema de medidas acústicas que iguale o supere las prestaciones de los sonómetros existentes en el mercado.

SonoPhone permitirá, mediante la conexión de un micrófono de medida adecuado, la realización de medidas de acuerdo a las normas técnicas en vigor, así como la posibilidad de programar, configurar y almacenar o trasmitir las medidas realizadas, que además estarán geolocalizadas con el GPS integrado en el dispositivo móvil, También se permitirá enviar los datos de la medida a un almacenamiento remoto en la nube.

La aplicación tiene una estructura modular en la que un módulo de adquisición de datos lee la señal del micrófono, un back-end efectúa el procesado necesario, y otros módulos permiten la calibración del dispositivo y programar y configurar las medidas, así como su almacenamiento y transmisión en red. Una interfaz de usuario (GUI) permite visualizar las medidas y efectuar las configuraciones deseadas por el usuario, todo ello en tiempo real.

Además de implementar la aplicación, se ha realizado una prueba de funcionamiento para determinar si el *hardware* del iPhone es adecuado para la medida de la presión acústica de acuerdo a las normas internacionales.

Sound pressure measurement is an extremely important process in the field of acoustic engineering, with applications in numerous subfields, like for instance building acoustics and noise control, where it is necessary to be able to accurately measure sound pressure in very diverse (and sometimes adverse) conditions.

On the other hand, the growing ubiquity of mobile devices such as smartphones or tablets, which combine processing power, connectivity, interactivity and an intuitive interface in a small size, makes it possible to use these devices as quality low-cost measurement systems.

This Project aims to use the input-output capabilities of iOS-based mobile devices, in particular the iPhone, together with their processing power, wireless connectivity and geolocation features, to implement an acoustic measurement system that rivals the performance of existing devices.

SonoPhone allows, with the addition of an adequate measurement microphone, to carry out measurements that comply with current technical regulations, as well as programming, configuring, storing and transmitting the results of the measurement. These measurements will be geolocated using the integrated GPS, and can be transmitted effortlessly to a remote cloud storage.

The application is structured in modular fashion. A data acquisition module reads the signal from the microphone, while a back-end module carries out the necessary processing. Other modules permit the device to be calibrated, or control the configuration of the measurement and its storage or transmission. A Graphical User Interface (GUI) allows visual feedback on the measurement in progress, and provides the user with real-time control over the measurement parameters.

Not only an application has been developed; a laboratory test was carried out with the goal of determining if the hardware of the iPhone permits the whole system to comply with international regulations regarding sound level meters.

A Pilar Prats
y Aurelio Domínguez
in memoriam

Índice

Índice de figuras

Índice de Tablas

Listados de código

Capítulo 1

Objetivos del Proyecto

1.1. Estructura de la memoria

Este Proyecto pretende dar solución a uno de los principales problemas prácticos de la ingeniería acústica, a saber, la medición del nivel de presión sonora de forma fiable, precisa y sencilla, haciendo uso de las posibilidades que ofrece uno de las tecnologías de más rápida evolución y que más han cambiado el mundo en la última década: los teléfonos inteligentes o smartphones.

En esta introducción se presenta la problemática que se pretende abordar, y cuáles son los objetivos a los que se pretende llegar. La parte I del proyecto está dedicada a las Bases Teóricas. En el capítulo 2 se estudia el estado actual de las tecnologías que han proporcionado o proporcionan soluciones prácticas, detallando los retos que presenta la medición del nivel de presión sonora, y dando una somera visión de su evolución histórica. También se hará una somera presentación de la plataforma sobre la que se desarrolla el proyecto, iOS, y porqué se ha elegido para este Proyecto.

Las bases teóricas se completan con el capítulo 3, que se centra en la principal regulación técnica en lo relativo a sonómetros: la norma internacional ISO/IEC 61672, con especial énfasis en su primera parte, titulada "Electroacústica. Sonómetros - Especificaciones". Esta norma detalla todo lo relativo a las funciones y requisitos de calidad de un aparato de medida de presión sonora que pueda utilizarse para mediciones de acuerdo con las normas internacionales en materia de acústica, control de ruido, etc.

El desarrollo del proyecto forma la parte II. El capítulo 4, se centra en la aplicación `SonoPhone`. Se presenta la estructura general de la aplicación, se explora el funcionamiento

de sus diferentes módulos y cómo éstos están relacionados entre sí. Se discuten diversos algoritmos usados por la aplicación para realizar ciertos cálculos y cómo se han implementado e integrado. También se define la interfaz de usuario (UI), y su adecuación a los principios de la interacción con el usuario. Finalmente se estudia cómo la aplicación propuesta da respuesta a los objetivos de diseño planteados.

La evaluación del hardware del iPhone para determinar si es adecuado para la implementación de un sistema profesional ocupa el capítulo 5. En él, se presentan los resultados de las pruebas realizadas y su interpretación.

Las conclusiones alcanzadas ocupan el capítulo 6. Asimismo se discutirán posibles desarrollos futuros de la aplicación, y su integración en proyectos más amplios.

El presupuesto del proyecto se encuentra detallado en el capítulo 7.

Los *apéndices* introducen aspectos interesantes encontrados en el proceso de realización del Proyecto, que aunque no guardan una relación directa con la solución al problema planteado, merecen cierta atención.

El apéndice A trata del concepto de *incertidumbre de medida*. De este concepto metrológico, importante para la aplicación de muchas de las especificaciones mencionadas en el capítulo 3, se proporciona una somera explicación para su comprensión, así como referencias bibliográficas para el lector interesado.

En el apéndice B se presenta la plataforma iOS, sobre la que se desarrolla el proyecto, con especial énfasis en cómo se realiza la adquisición y procesado de audio. Se introducen asimismo diversos conceptos necesarios para entender el desarrollo de aplicaciones, tales como la arquitectura modelo-vista-controlador (MVC), y las peculiaridades del entorno de desarrollo y el lenguaje de programación Objective-C.

La implementación práctica de las pruebas de evaluación de hardware que se discuten en el capítulo 5 ocupa el apéndice C. En él se detallan las bases teóricas del método utilizado, el procesado necesario y la comparación con el objetivo dictado por la Norma.

En el apéndice D se trata un aspecto práctico para la realización de cualquier Proyecto en iOS, que todo desarrollador debe conocer; el funcionamiento de los distintos *Developer Program* que Apple pone a disposición de particulares y organizaciones. En concreto, se detalla el procedimiento para afiliarse al programa al que la UPM está suscrita, para poder probar la aplicación en un dispositivo.

Presentar Amazon S3, la infraestructura de almacenamiento en la nube que utiliza SonoPhone, es el objetivo del apéndice E. En él se recorre paso a paso el proceso de darse

de alta en el servicio, poner en marcha el almacenamiento e integrarlo en una aplicación iOS.

1.2. Objetivos del proyecto

La medición de la presión sonora es uno de los problemas centrales de la ingeniería acústica, constituyendo un proceso imprescindible para la evaluación y presentación de soluciones a problemas en todos los campos de esta disciplina. Desde el diseño de salas de cine hasta la evaluación del aislamiento que proporciona una pantalla acústica al paso de un tren, los sistemas de medición de la presión sonora han buscado proporcionar al ingeniero una herramienta sencilla y precisa para el diagnóstico de problemas acústicos y la comprobación in situ de los cálculos y simulaciones realizadas.

Sin embargo, no sólo es una herramienta útil para el especialista, sino que en el marco de una creciente preocupación social y política por los problemas de ruido, la medición de la presión sonora también tiene una gran importancia para los ciudadanos y el Estado, dado que una medición adecuada es imprescindible para hacer cumplir la legislación con respecto al ruido, cada vez más clara y restrictiva, que en los últimos años ha entrado en vigor, permitiendo tanto la detección de infracciones como la elaboración de planes de prevención de la contaminación acústica a largo plazo.

Por todo ello un sistema de medición de presión sonora adecuado deberá poseer las siguientes características:

Precisión: Se debe poder proporcionar una medida que resuma con exactitud las características del campo sonoro y permita la comparación con medidas en distintas condiciones y con otros aparatos. Se deben asimismo proporcionar unos márgenes de tolerancia claros y adecuados, así como unas condiciones de operación en las que se garantice que la medida se ajusta a las especificaciones internacionales relevantes. El sistema ha de estar correctamente calibrado (con calibración trazable a un laboratorio nacional) y debe ofrecer la posibilidad de ser recalibrado.

Fiabilidad: Las medidas deben ser repetibles. Hay que indicar con claridad las condiciones en las que fueron obtenidas. El sistema ha de ser capaz de operar en entornos con condiciones ampliamente variables sin sufrir pérdida de precisión. Es recomendable proporcionar una manera de almacenar las medidas realizadas para su futuro estudio.

Versatilidad: El sistema debe poder efectuar distintas medidas según los requerimientos del usuario, adecuándose a los estándares internacionales aceptados, e indicando en su manual cuáles son las diferencias entre los distintos tipos de medida.

Resistencia: Hay que asegurar que el sistema está preparado para ser manipulado en distintos entornos; debe ser fácil de transportar y desplegar, poseer cierta autonomía y ser razonablemente resistente a distintas condiciones meteorológicas. El fabricante ha de indicar los márgenes de temperatura, humedad, etc. bajo las que su producto funciona correctamente.

Facilidad de manejo: Se debe mostrar la información necesaria de forma clara y correcta, de manera que pueda comprobarse de un vistazo. Las opciones relativas a la medición deben poder cambiarse rápidamente. Los elementos de interacción con el usuario han de ser resistentes, claramente indicados y fáciles de localizar.

Coste bajo: Para ser competitivo, el producto debe ofrecer un paquete completo de funcionalidad a un precio menor que los competidores, en consonancia con la progresiva bajada de precios que el avance de la tecnología lleva aparejado.

Por otro lado, el fuerte desarrollo de las plataformas móviles y su creciente ubicuidad las convierten en un campo de experimentación muy vivo para el desarrollo de aplicaciones que cambien sustancialmente cómo se contempla la medida de la presión acústica, permitiendo que deje de ser un campo reservado a los especialistas de la Acústica.

Por tanto, la razón de ser de este Proyecto es el desarrollo de un sistema que cumpla con los objetivos enumerados al tiempo que aporte una innovación, no limitándose a reproducir la funcionalidad de los sistemas de medida de la presión sonora ya existentes, sino buscando introducir nuevas capacidades propias de una plataforma tan potente, versátil y ampliamente utilizada como iOS.

Pero no solo basta con el desarrollo, sino que es necesario realizar una serie de pruebas de funcionamiento para comprobar si SonoPhone puede competir en igualdad de condiciones con los sistemas de medida de presión acústica existentes en el mercado. El objetivo debe ser el cumplimiento de las especificaciones recogidas en la norma IEC 61672, por lo menos para la clase 2. Si se logra cumplir con estas especificaciones, se podría abrir un campo de posibilidades que dejara obsoletos a los sonómetros actuales.

1.2.1. Características de la aplicación

Sonophone permitirá:

- Medir niveles de sonido instantáneos y promediados en el tiempo, ajustándose en lo posible a la norma IEC 61672.
- Seleccionar el tipo de medición que se desea.
- Almacenar información (metadatos) respecto a las medidas: fecha y localización, entre otros.
- Almacenar el resultado de las mediciones en el teléfono, permitiendo descargarlas a un ordenador.
- Transmitir las mediciones a un servidor remoto en la nube, con el propósito de disponer de una base de datos de medidas con vistas a la elaboración de mapas de ruido o minería de datos.

Parte I

Bases Teóricas

Capítulo 2

Bases teóricas

A continuación se hará una revisión de los sistemas existentes junto con una breve perspectiva histórica de su desarrollo, y se expondrán las ventajas que las plataformas móviles, y más concretamente las basadas en iOS, tienen para el desarrollo de un sistema innovador.

2.1. Revisión de los sistemas existentes

Tradicionalmente los sistemas de medición de la presión sonora se han clasificado grosso modo en dos tipos:

Sonómetros integrados pensados con la idea de poder llevarse en la mano durante las mediciones de campo, con un display integrado que permite visualizar la medición realizada y algún tipo de control que permite seleccionar cómo se realiza dicha medición.

Sonómetros distribuidos, que constan de varias partes interconectadas con funcionalidad independiente. Sacrifican parte de la movilidad y resistencia de los sistemas integrados a cambio de permitir mayor flexibilidad y control sobre las medidas y cómo estas se presentan o almacenan.

Ambos tipos de sistemas constan de distintos módulos, como muestra la figura 2.1.

Módulo de adquisición: se encarga de convertir la presión acústica en una señal que pueda ser procesada. Por tanto, consta de un transductor (micrófono), con su corres-

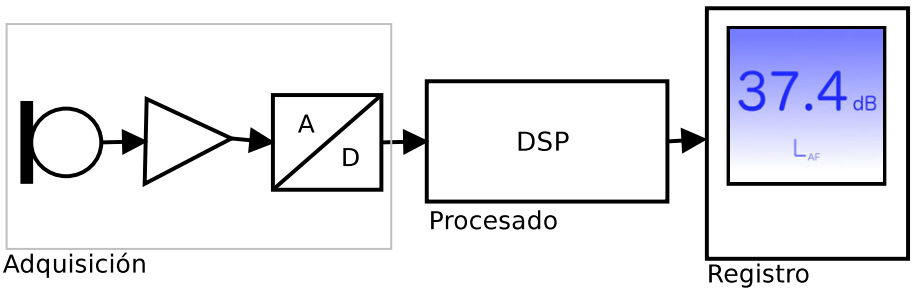

Figura 2.1: Diagrama de bloques de un sonómetro digital típico.

pondiente preamplificador, y en el caso de que el procesado subsiguiente sea digital, un convertidor analógico-digital (ADC).

Módulo de procesado: analiza la información proporcionada por el módulo de adquisición, efectuando los filtrados, correcciones y cálculos que sean necesarios, de acuerdo con la configuración que desee el usuario.

Módulo de registro: se encarga de recoger los resultados del procesado y mostrarlos al usuario mediante alguna clase de display que los muestre con claridad para que éste pueda interpretarlos. Opcionalmente, los resultados podrán ser almacenados en una memoria o enviados a un dispositivo externo para su registro o procesado adicional.

2.1.1. Sonómetros

Según el tipo de medidas que realizan, hablaremos de sonómetros

Convencionales: miden instantáneamente el nivel de presión sonora con promediación temporal exponencial, usando una constante de tiempo especificada.

Integradores-promediadores: miden niveles de sonido promediados en un periodo de tiempo.

Integradores: miden niveles de exposición sonora.

Estos tipos de sonómetros se definen en la norma IEC 61672 (UNE-EN, 2005).

No obstante, la tendencia ha sido la de que un solo instrumento realice todas estas funciones, incluso incorporando capacidades adicionales como por ejemplo el análisis espectral

del sonido medido, gracias a la mayor versatilidad de la tecnología digital en la que están realizados la práctica totalidad de los sonómetros actuales.

Los sonómetros analógicos realizaban el procesado mediante circuitería electrónica delicada, propensa a cambiar sus características con la temperatura y el tiempo y difícil de ajustar con precisión, además de depender de displays tipo galvanómetro cuya mecánica introduce un integrado adicional. No obstante, su principal desventaja consistía en su escasa versatilidad.

En contraste, los sonómetros digitales basan su operación en procesadores digitales de señal, que tratan la señal acústica como secuencia digital, realizando operaciones aritméticas y acumulando valores en sus registros. Además esta tecnología permite el almacenamiento de resultados en memorias tanto volátiles (tipo RAM) como permanentes, y la interconexión con otros dispositivos de medida, ordenadores, etcétera, con las consiguientes ventajas en la operatividad del sistema.

2.1.2. Sistemas integrados

Los sistemas integrados han tomado relevancia con el desarrollo de los ordenadores personales, que son ya capaces de manejar entrada y salida de audio de alta calidad y de procesar los datos en tiempo real. Los sistemas integrados se realizan con paquetes de *software* que no dependen de una configuración específica, sino que permiten su uso en una gran variedad de situaciones de medida. Un ejemplo de este tipo de sistemas es el ofrecido por el fabricante ACOEM, que se muestra en la figura 2.2. Hay que destacar que para la norma IEC 61672, tal como se desarrolla en el capítulo 3, a esta clase de sistemas se le puede aplicar la consideración de sonómetro siempre y cuando el conjunto de hardware y software cumpla con las especificaciones de la Norma.

2.2. Medida de la presión sonora

La presión se define como la razón de la fuerza ejercida por la superficie donde se ejerce. Se mide en el Sistema Internacional en newtons por metro cuadrado ($\mathrm{N\,m^{-2}}$), unidad conocida como Pascal (Pa).

La presión ambiental es la presión que ejerce un fluido en equilibrio hidrostático. De acuerdo con el principio de Pascal, esta presión se ejerce con igual intensidad en todas las direcciones. La presión atmosférica varía con la temperatura, humedad, altitud y otros

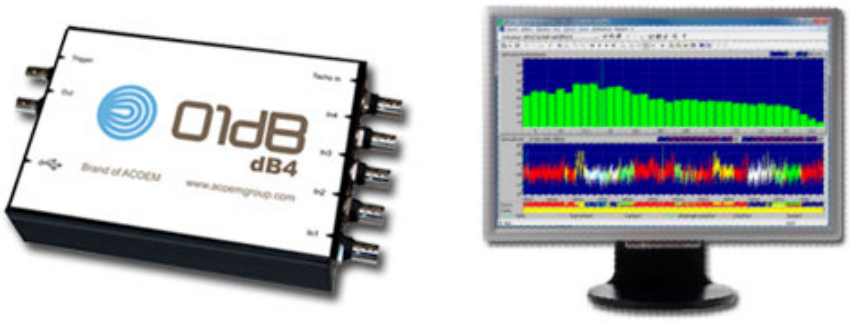

Figura 2.2: Analizador dB4 y software dBTRAIT del fabricante ACOEM

factores, siendo la presión estándar, establecida como una atmósfera (1 atm) la de 101.325 kPa.

La presión sonora se define como la variación de la presión con respecto a la presión ambiental (atmosférica, en el caso del aire) causada por una onda sonora. La característica principal del sonido es que las perturbaciones de presión se transmiten rápidamente por el espacio a la llamada velocidad del sonido (Fahy, 2003).

La ecuación de propagación del sonido

$$\frac{\partial^2 p}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = 0 \tag{2.1}$$

Determina la dinámica de las ondas sonoras. La velocidad de *propagación* de las ondas sonoras depende de la presión barométrica P_0, del coeficiente adiabático γ y de la densidad del fluido ρ_0 mediante la siguiente ecuación

$$c = \sqrt{\left(\frac{\gamma P_0}{\rho_0}\right)} \tag{2.2}$$

Extraída de (Kuehn, 2009), que arroja un valor de $331.6\,\mathrm{m\,s^{-1}}$ a $0\,^{\circ}\mathrm{C}$. c varía con las condiciones ambientales, especialmente con la temperatura. La presión sonora, pues, es la desviación instantánea de la presión con respecto a la presión ambiente: una compresión corresponde a una desviación positiva y una rarefacción a una desviación negativa. Medir la presión sonora implica medir la magnitud de dicha desviación.

Lo que hace que el sonido sea considerado como tal es el que provoque una sensación auditiva. Los límites de dicha sensación auditiva se suelen acotar en frecuencia, como el rango entre 20 Hz y 20 kHz y en nivel, siendo el límite inferior de unos 20 µPa (micropascales, millonésima parte de un Pascal), mientras que el superior, llamado *Umbral de dolor*,

no está claramente definido aunque puede superar las decenas de kPa (kilopascales, miles de Pascales)

Este rango de variaciones, de unos 10 órdenes de magnitud en nivel, hace necesaria la adopción de una escala logarítmica, y que por tanto no hablemos de presión sonora per se sino de nivel de presión sonora. El nivel de presión sonora L_p, también conocido por sus siglas en inglés SPL (Sound Pressure Level) se define como 10 veces el logaritmo de la razón entre el cuadrado de la presión medida y la presión de referencia. Esta presión de referencia se establece como la de un tono puro de 1 kHz en el umbral de audibilidad, igual a 20 micropascales (µPa).

$$L_p = 10 \log \frac{p^2}{(2 \cdot 10^{-5}\text{Pa})^2} \,\text{dB} \tag{2.3}$$

Un análisis de la fórmula nos revela que el nivel de presión sonora aumenta en 6 dB cada vez que se dobla el valor de la presión, y 20 dB cada vez que se centuplica ésta. Esto permite comprimir el rango de valores con los que se trabaja. Además, la elección del umbral de percepción como valor de referencia asegura que todos los niveles medidos sean positivos.

En la fórmula se ha representado la presión medida con un símbolo genérico, p, que esconde la dependencia espaciotemporal de la presión sonora, que en general se expresaría como $p(\vec{\mathbf{r}}, t)$. Elevando el valor instantáneo al cuadrado, tenemos en cuenta el hecho de que la presión puede tomar valores negativos. Aplicando las propiedades de los logaritmos, se puede llegar a una expresión alternativa

$$L_p = 20 \log \frac{p}{2 \cdot 10^{-5}\text{Pa}} \,\text{dB} \tag{2.4}$$

Que sin embargo pierde la información de *polaridad* de la presión, suponiendo que esta es siempre positiva y por tanto no puede aplicarse a la presión instantánea. Implícitamente, supone por tanto que se realiza una *integración* de la presión sonora previa al cálculo de L_p. En la sección 2.2.1 discutiremos las distintas técnicas para realizar esta integración.

El sonido presenta la peculiaridad de definirse en relación a una percepción subjetiva. El sistema auditivo humano presenta numerosos efectos no lineales, de enmascaramiento, etcétera, de cuyo estudio se ocupa la psicoacústica. Uno de los efectos más importantes es la respuesta en frecuencia no plana y dependiente del nivel del sonido que presenta el oído. En ese sentido, se intenta mejorar la correlación entre el nivel medido y la percepción de

la sonoridad de dicho sonido mediante la aplicación de redes de ponderación frecuencial, que se tratan en la sección 2.2.2.

De la medida del nivel de presión acústica se extraen unos valores estandarizados que pretenden ser útiles al ingeniero para caracterizar el sonido que mide, y que debe elegir según el propósito que persiga. Este apartado se tratará en la sección2.2.3.

Cabe destacar que la presión no es la única magnitud acústica relevante, por ejemplo puede medirse la intensidad o la velocidad de las partículas (Brixen, 2012). Sin embargo, la medida de la presión es la más extendida.

2.2.1. Integración

Para señales estacionarias, la potencia de la señal permanece constante con el tiempo. Sin embargo, en señales reales, tales como las que resultan de medidas, solo se puede asumir la estacionariedad de la medida en cortos periodos de tiempo. Solo puede producirse una medida coherente y comparable en relación a un tiempo en el que se supone que las propiedades estadísticas permanecen constantes, puesto que en la práctica la obtención de la medida se realiza tomando en cada instante las muestras anteriores hasta el tiempo de promediado, T, elegido. La elección de T influirá en la medida obtenida: una serie de medidas de potencia o nivel obtenida con un T largo tenderá a suavizar las variaciones bruscas y a resaltar la tendencia general de los datos, mientras que una con T pequeño registrará correctamente los picos y valles.

Además, no solo es relevante la elección de un tiempo de promediado, sino que existen dos métodos para realizar dicho promediado que también influirán en la medida:

Promediado lineal: Se da a todas las muestras temporales el mismo peso.

Promediado exponencial: Se aplica a las muestras más recientes mayor peso que a las pasadas. Esta operación se discutirá con detalle en la sección 3.4.4.

La ponderación temporal encuentra aplicación en los casos en los que el sonómetro esté funcionando en modo convencional (ver sección 2.1.1). Cuando se promedia la energía, no tendría sentido considerar estos periodos cortos de tiempo.

2.2.2. Ponderación en frecuencia

Desde que Fletcher y Munson describieran sus curvas de sonoridad constante, se sabe que la respuesta del sistema auditivo no es lineal. En general se perciben con más sensibi-

lidad los sonidos en torno a 3 kHz, mientras que las frecuencias situadas a ambos extremos del espectro necesitan más nivel para ser percibidas con la misma sonoridad por el oyente.

Para intentar contrarrestar ese efecto y hacer que las medidas obtenidas por distintos sistemas sean comparables, se introdujo la ponderación A, que introduce un filtrado paso alto y paso bajo con características estandarizadas. La presión acústica medida se hace pasar primero por este filtro, antes de cualquier procesado posterior. Al nivel de presión sonora con ponderación A se lo suele denotar como dBA.

Un inconveniente de la ponderación A es que puede minusvalorar la contribución a la energía total de la baja frecuencia. Esto es importante para el control de ruido, dado que el ruido de baja frecuencia se considera más molesto.

Otra ponderación digna de mención es la C, igual que la A pero sin el filtrado paso alto. Existen otras ponderaciones, como la B o la I, que sin embargo no tienen aplicación hoy en día.

2.2.3. Medidas adecuadas

El objetivo de cualquier medida es que sea útil para el propósito para el que se realiza, ya sea comprobar que un sistema funciona según lo especificado, diagnosticar un problema o hacer cumplir una especificación o regulación.

Así, por ejemplo, cuando se trata de determinar la exposición al ruido, importa saber la energía total que percibe el oyente, y por tanto se debe integrar la energía acústica en un periodo de tiempo largo. Sin embargo, si lo que se pretende es registrar la aparición de eventos acústicos de tipo impulsivo, se debe disponer de una historia de niveles medidos con tiempos de integración muy cortos.

Es de especial importancia el nivel equivalente L_{Aeq}, que se define como la media energética del nivel de ruido promediado en el intervalo de tiempo de medida, o el nivel que tendría un sonido continuo de la misma energía y duración de la medida. El nivel de exposición sonora (SEL) se define como el nivel equivalente a un segundo.

2.3. Plataformas móviles

En los últimos años han cobrado gran relevancia las plataformas móviles. Los teléfonos inteligentes o *smartphones*, descendientes directos de los PDA, y las tabletas han experimentado un extraordinario incremento de ventas en todo el mundo. La consultora Gartner

estima que se vendieron más de 200 millones de *smartphones* en el primer cuarto del año fiscal 2013. En cuanto a las tabletas, IDC calcula que en ese año se venderán 229 millones de unidades, llegando a superar las ventas de ordenadores portátiles.

Los dispositivos móviles presentan diversas características que los hacen interesantes para aplicaciones como la que concierne a este Proyecto:

Portabilidad Los smartphones están diseñados para llevar en el bolsillo, y las tabletas suelen tener las dimensiones de un cuaderno delgado. Esto permite que el usuario los lleve encima prácticamente en todo momento.

Conectividad Buscan una conectividad ubicua, mediante el uso de WiFi, redes móviles y otras tecnologías como Bluetooth o NFC. Además permiten a las aplicaciones variar su funcionamiento dependiendo de la ubicación del usuario, que se puede obtener a partir de las redes a las que esté conectado el aparato o mediante el uso de GPS. Los dispositivos pueden estar dotados de diversos sensores y actuadores, tales como micrófono, altavoz, entrada y salida de audio, acelerómetro, giroscopio y magnetómetro, entre otros.

Interactividad Su característica principal es la pantalla táctil, que ofrece una interacción mucho más directa con el software, sin la mediación de aparatos como el ratón o teclado.

Potencia de procesamiento Los avances tecnológicos en la fabricación de microchips permiten que se utilicen en estos dispositivos los llamados *System on a Chip* (SoC), que integran procesadores de dos y hasta cuatro núcleos, GPUs y memoria caché a la altura de los de un ordenador portátil, además de abundante memoria RAM (alrededor de 1GB). La duración limitada de las baterías obliga al sistema operativo a ser muy eficiente en el uso de los recursos.

Énfasis en la pantalla La superficie útil de los *smartphones* y tabletas está ocupada en su práctica totalidad por la pantalla táctil, con los botones ocupando una posición y función secundaria. Esto condiciona cómo se desarrollan aplicaciones y da al dispositivo un aspecto de *tabula rasa* que el desarrollador de aplicaciones llena totalmente con su creación.

El desarrollo para dispositivos móviles se centra en aplicaciones (apps) autocontenidas

diseñadas para uno de los sistemas operativos móviles usando un SDK[1] adecuado para la plataforma y distribuidas mediante la tienda de aplicaciones correspondiente. Para ello, el desarrollador debe estar inscrito en la tienda (pagando una suscripción), y las aplicaciones deben superar una revisión por parte de la misma.

A continuación se enumeran los sistemas operativos móviles más importantes actualmente:

iOS Desarrollado por Apple para sus dispositivos: iPhone, iPad e iPod Touch. Actualmente en su versión 6.1, con la versión 7 ya anunciada. Está basado en el sistema operativo de escritorio Mac OS X.

Android Sistema de código abierto basado en Linux, desarrollado por Google y apoyado por la Open Handset Alliance (OHA), una asociación de fabricantes de hardware y software y compañías telefónicas. Es el sistema operativo móvil más usado, capaz de correr en una variedad enorme de dispositivos.

Windows Phone Desarrollado por Microsoft, es el más reciente competidor en este mercado. Está diseñado para la convergencia, en funcionalidad y código, con Windows 8 para sistemas de escritorio.

Otros Por ejemplo, BlackBerry OS para los dispositivos de este fabricante, o sistemas de código abierto como Ubuntu Touch o Firefox OS.

2.3.1. La plataforma iOS

Al elegir la plataforma iOS para el desarrollo de este Proyecto, se tuvieron en cuenta las ventajas e inconvenientes de ésta respecto a las mencionadas antes, en especial Android.

Entre las ventajas cabe destacar principalmente que al elegir iOS se elige también el dispositivo que se utiliza: iPhone o iPad. Esto implica:

- Mejor acoplamiento del sistema operativo al hardware. Esto es muy importante para aplicaciones que procesan audio, como SonoPhone, ya que una API de audio más optimizada implica una latencia menor. En contraste, la interfaz de bajo nivel de un dispositivo Android depende en gran medida del fabricante, y la alta prioridad que requiere el audio de baja latencia puede interferir con las estrategias para ahorro de batería del terminal.

[1] *Software Development Kit*, kit de desarrollo de software.

- Seguridad de que el sistema operativo está actualizado. En contraste, en Android una gran masa de dispositivos utilizan versiones desactualizadas del SO.
- Estado del arte. Existen numerosas aplicaciones de audio de baja latencia para iOS. Las APIs están bien documentadas y probadas y existe una importante comunidad de desarrolladores de audio.

Entre las desventajas cabría destacar que no se llega a la gran mayoría de dispositivos, lo restrictiva que es la plataforma y la curva de aprendizaje que implica tener que aprender un nuevo lenguaje de programación, Objective-C (a menos que se cuente con experiencia en desarrollo para OS X). En contraste, las aplicaciones Android se escriben en Java, y las Windows Phone en .NET, dos de los lenguajes más usados.

A grandes rasgos, el desarrollo en iOS implica:

- Utilizar el SDK de iOS. Esto implica el uso de XCode, el entorno de desarrollo (IDE) de Apple, para la compilación, depuración, etc. XCode solo está disponible para OS X. Por tanto,*sólo se puede desarrollar para iOS desde un ordenador Mac.* Además, XCode incluye toda la documentación oficial de iOS.
- Suscribirse a un Developer Program. Por un precio de 99 dólares al año, se permite distribuir la aplicación en la App Store y probarla en un dispositivo real. Para uso académico, existe la opción de usar gratuitamente el University Program asociado a la institución[2]. Sin suscripción a un Developer Program, solo se puede probar la aplicación en el simulador iOS que provee el SDK.
- Seguir las prácticas recomendadas, en cuanto a código e interacción con el usuario. Apple publica en sus *Human Interface Guidelines* y *App Programming Guide* los principios de diseño que debe seguir una app, que se tratarán en más detalle en la sección 2.3.3.

2.3.2. Estructura del SDK

El conjunto de tecnologías, agrupadas en frameworks, que forman el SDK de iOS se estructura en capas, siguiendo un nivel ascendente de abstracción. Así, las capas de nivel inferior están más cercanas al hardware, y las de nivel superior permiten interactuar a

[2]Para este Proyecto, se creó un perfil en el University Program de la UPM. Se puede encontrar una introducción práctica al University Program en el anexo D.

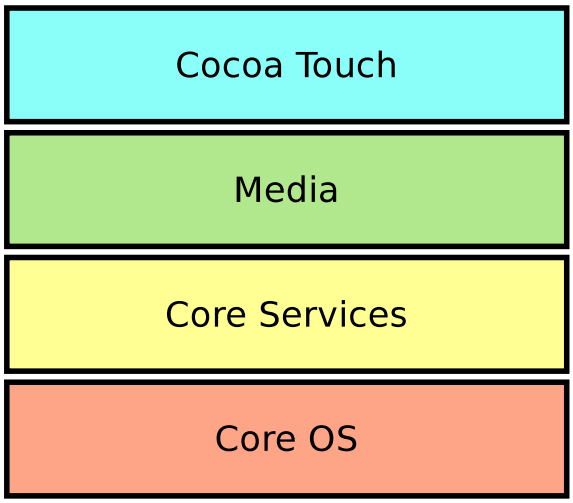

Figura 2.3: Arquitectura del SDK de iOS

un nivel más cercano al usuario, permitiendo escribir menos líneas de código y evitando al desarrollador lidiar con constructos complejos como hilos de ejecución o sockets. Esta arquitectura se ilustra en la figura 2.3.

Cocoa Touch Contiene los servicios básicos de la interfaz de usuario (UI) de la aplicación, entre ellos los elementos visuales elementales (botones, cuadros de texto, etc.), plantillas predefinidas como tablas o barras de herramientas, interacción con servicios del sistema como fotos, calendario o mapas, así como el reconocimiento de gestos táctiles (arrastrar, pulsar, rotar...) y otros servicios.

Media Agrupa a las interfaces de programación relativas a audio y vídeo, tanto de alto como bajo nivel. Éstas permiten tanto la reproducción simple de archivos como la manipulación sofisticada de flujos de audio y vídeo, además de proporcionar acceso a tecnologías pensadas para juegos como OpenGL y OpenAL.

Core Services Proporciona las interfaces básicas de programación, tales como colecciones de objetos, cadenas de caracteres, etc, así como funciones de red, seguridad...

Core OS Contiene librerías que interactúan directamente con el sistema operativo y otras librerías para el procesado de señales o encriptación.

2.3.3. Principios de diseño de una app

Apple, mediante la propia arquitectura de sus frameworks, la publicación de sus diversas guías (como las *Human Interface Guidelines*) y el proceso de revisión que hace en la App Store, promueve que las apps sigan determinados principios de diseño. Esto tiene el objetivo de que la experiencia del usuario sea uniforme en todo el sistema operativo, con el fin de que las apps sean totalmente intuitivas. En iOS, las aplicaciones no tienen manual de usuario.

Estos principios comprenden tanto prácticas de programación como de diseño. Como ejemplo de las primeras podríamos citar el uso del lenguaje de programación Objective-C y el framework Cocoa Touch, que favorecen el uso de patrones como Modelo-Vista-Controlador (MVC), delegación y protocolos o codificación clave-valor (KVC). Por otra parte, el desarrollador está limitado a las APIs públicas que Apple ofrece; el uso de APIs no documentadas es motivo de rechazo en la App Store.

Otros principios de diseño son consecuencia de las limitaciones que impone la plataforma, sobre todo en cuanto a espacio en pantalla y conservación de la batería. Por ejemplo, a diferencia de un SO de escritorio, solo hay una "ventana" activa. Por tanto se debe evitar sobrecargar el aspecto visual de la aplicación, siendo preferible dividir en distintas vistas entre las que se navega, estilo por otra parte impuesto por los controladores predefinidos de iOS.

La asignación limitada de recursos implica que solo una aplicación está activa a la vez, y debe estar preparada para perder ese protagonismo en cualquier momento debido a que el usuario cambie de aplicación o reciba una llamada. Siempre se debe hacer un uso eficiente de los recursos, sobre todo aquellos que gastan más batería, como conexiones a la red, GPS o cálculos intensivos. Además, la aplicación no debe jamás quedar un estado en el que no responda; se han de usar técnicas de computación asíncrona para no bloquear la interfaz de usuario.

El usuario de iOS tiene unas expectativas de interacción; es decir, espera poder usar una aplicación nueva de forma similar a las que ya ha utilizado. Esto debe aprovecharlo el desarrollador para proporcionar una mejor experiencia. Un ejemplo es el uso de los gestos táctiles, que permiten la manipulación directa de los objetos de la aplicación. No se debe imponer al usuario una manera de usar la aplicación, sino aprovechar su instinto.

Otros principios de diseño conciernen los objetivos de la aplicación. Por ejemplo, no se recomienda desarrollar aplicaciones que hagan muchas cosas, sino centrarse en proporcionar menos características y optimizar al máximo el rendimiento. De todos modos la plataforma

no favorece aplicaciones complejas, y en el entorno competitivo de la App Store es más fácil sobresalir dando respuesta a una necesidad específica.

El aspecto visual es tan importante en iOS como la funcionalidad de la aplicación. Apple pone gran énfasis en su recomendación de proporcionar gráficos personalizados: iconos, texturas, etc. y promueve que se aprovechen las *metáforas de diseño* y el "skeuomorfismo"[3], con el fin de explotar la familiaridad del usuario con aplicaciones ya existentes o reproducir usos del "mundo real".

2.4. Principios de desarrollo del Proyecto

Una vez examinados los sistemas de medida existentes, los principios teóricos en los que ha de basarse el Proyecto y la plataforma software en la que se implementa, es necesario planear cómo se llevará a cabo el Proyecto.

El uso de la plataforma iOS queda justificado si se pueden cumplir los objetivos de un sistema de medida adecuado enumerados en la sección 1.2. Incluso si no se consigue cumplir la norma IEC 61672, que haría de Sonophone un sonómetro certificado, la aplicación sería innovadora al poder ofrecer, en una plataforma ampliamente utilizada, capacidades de instrumentos profesionales, especializados y costosos, además de poder aprovechar la conectividad del iPhone para proporcionar funcionalidades de red inéditas hasta ahora.

En cuanto a la estructura en bloques del sistema de medida, tal y como se describe en la sección 2.1, el rol de bloque de adquisición lo formaría la entrada de audio del iPhone, bien por el micrófono integrado, bien por la entrada de micrófono. El desarrollador no tiene control sobre este bloque, ya que el funcionamiento de estas partes del teléfono forma parte de las especificaciones privadas de Apple. El bloque de procesado corresponde a las APIs de audio y procesado de señal de iOS, que toman el audio digitalizado y lo procesan adecuadamente. Las APIs de interfaz de usuario de Cocoa Touch permiten mostrar el resultado de los cálculos y seleccionar el tipo de medida que se desee hacer, actuando como bloque de registro.

[3]Neologismo que denota la similaridad de un diseño con un objeto familiar. Por ejemplo, una aplicación para tomar notas puede utilizar texturas que sugieran papel y una fuente similar a la escritura a mano.

Capítulo 3

Norma UNE-EN 61672-1

3.1. Introducción

La principal norma internacional que regula los sonómetros es la IEC 61672:2002, que ha sido armonizada por AENOR (UNE-EN, 2005). Esta norma tiene tres partes.

Parte 1: Especificaciones Parte principal de la norma en la que se detallan todas las especificaciones técnicas que debe cumplir un sonómetro especificado.

Parte 2: Ensayos de evaluación de modelo Define los ensayos que se deben realizar para comprobar que un sonómetro cumple las especificaciones definidas en la parte 1.

Parte 3: Evaluaciones periódicas Define los ensayos que aseguran al usuario la conformidad con la norma de su sonómetro bajo las condiciones de referencia.

En este capítulo nos centramos en la parte 1, por ser la que define las especificaciones a las que debe ajustarse el sistema.

3.2. Objeto y campo de aplicación

La norma se aplica a tres tipos de instrumentos de medida del sonido:

Sonómetros convencionales capaces de medir el nivel de presión sonora con ponderación temporal exponencial.

Sonómetros integradores-promediadores capaces de medir el nivel continuo equivalente.

Sonómetros integradores que miden niveles de exposición sonora.

En realidad esta distinción es sobre todo teórica, ya que muchos instrumentos actuales son capaces de realizar todas estas medidas e incorporar capacidades adicionales (como por ejemplo análisis en frecuencia).

Se definen dos clases de cumplimiento de la norma. Los sonómetros de **Clase 1** tienen unas tolerancias más restrictivas que los de **Clase 2**, siendo por lo demás sus especificaciones exactamente iguales.

La norma puede aplicarse tanto a sistemas de mano autocontenidos con micrófono montado y una pantalla para la presentación de resultados como a sistemas formados por diversos subsistemas conectados, con procesadores de señal analógicos o digitales y tantos dispositivos como sean necesarios para su operación como instrumento de medida.

Asimismo, los sistemas que cumplen la norma pueden estar tanto diseñados para medir en presencia de un operador como para operar autónomamente, aunque las especificaciones acústicas se definen sin la presencia del operador.

3.3. Definiciones

En esta sección la norma define todos los términos necesarios para su comprensión y aplicación. Algunos de los términos más importantes que aparecen son:

Nivel de presión acústica Veinte veces el logaritmo decimal del cociente entre el valor cuadrático medio de una presión dada y la presión acústica de referencia, definida ésta como igual a 20 µPa para sonido aéreo.

Ponderación frecuencial Diferencia en decibelios entre el nivel que el sonómetro muestra en su dispositivo de presentación de resultados y el nivel correspondiente a una señal sinusoidal de amplitud constante, expresado como función de la frecuencia.

Ponderación temporal Función exponencial temporal que pondera el cuadrado de la presión acústica medida.

Nivel sonoro ponderado temporalmente Nivel de presión acústica obtenido con una

ponderación frecuencial y temporal normalizadas. Es el nivel que miden los sonómetros convencionales. Se define matemáticamente como

$$L_{A\tau}(t) = 20\log\left\{\left[\frac{1}{\tau}\int_{-\infty}^{t} p_A^2(\xi)e^{-\frac{(t-\xi)}{\tau}}\mathrm{d}\xi\right]^{1/2} \Big/ p_0\right\} \tag{3.1}$$

donde

- τ es la constante de tiempo de la ponderación temporal en segundos,
- ξ es una variable de integración desde algún instante en el pasado hasta el tiempo de observación t,
- $p_A(\xi)$ es la presión acústica con ponderación frecuencial A, y
- p_0 es la presión acústica de referencia.

Esta ecuación expresa un proceso que se ilustra en la figura 3.1

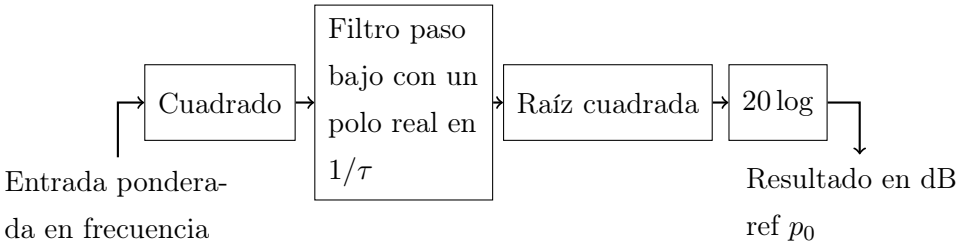

Figura 3.1: Diagrama de bloques para la obtención del nivel con ponderación temporal

Nivel de presión acústica de pico Veinte veces el logaritmo decimal del cociente entre la presión máxima medida en un intervalo de tiempo establecido y la presión acústica de referencia. La norma especifica la medida de este nivel con ponderación C, L_{Cpk}.

Nivel de sonido continuo equivalente Nivel de un sonido de amplitud constante que contiene la misma energía que el medido en un tiempo especificado; es la principal magnitud que miden los sonómetros integradores-promediadores. Se define como veinte veces el logaritmo decimal del cociente de la presión cuadrática, con ponderación frecuencial normalizada, promediado en un intervalo de tiempo especificado, y la presión acústica de referencia. Matemáticamente,

$$L_{Aeq} = 20\log\left\{\left[\frac{1}{T}\int_{t-T}^{t} p_A^2(\xi)\mathrm{d}\xi\right]^{1/2} \Big/ p_0\right\} \tag{3.2}$$

Donde

- T es la duración del intervalo de promediado en segundos,
- ξ es una variable de integración sobre el intervalo de promediado que finaliza en el tiempo de observación t,
- $p_A(\xi)$ es la presión acústica con ponderación frecuencial A, y
- p_0 es la presión acústica de referencia.

Es interesante notar que, según esta definición, no se aplica ponderación temporal, puesto que lo que tiene sentido en esta medida es la *energía total* en el intervalo de tiempo especificado.

Exposición sonora Es la integral temporal del cuadrado de la presión acústica durante un intervalo de tiempo o la duración de un suceso establecidos. Matemáticamente,

$$E_A = \int_{t_1}^{t_2} p_A^2(t)\mathrm{d}t \tag{3.3}$$

Donde los límites de integración, t_1 y t_2, se definen por la duración del suceso.

La unidad de la exposición sonora en el SI es el Pascal cuadrado-segundo (Pa^2 s), si bien para ciertas aplicaciones es más usado el Pascal-cuadrado hora (Pa^2 h)[1]. El nivel de exposición sonora L_{AE} se define como veinte veces el logaritmo decimal del cociente de la exposición sonora y la exposición sonora de referencia, definida como $p_0^2 T_0$, donde p_0 es la presión acústica de referencia y $T_0 = 1\,\mathrm{s}$.

Magnitudes de referencia Se definen el *punto de referencia del micrófono*, la *dirección de referencia*, el *ángulo de incidencia del sonido*, medido con respecto a la dirección de referencia, y el *nivel de presión acústica de referencia*, respecto al cual se definen las ponderaciones frecuenciales, respuestas direccionales de referencia, etcétera, entre otros.

Tren de ondas Uno o más ciclos de una señal sinusoidal, empezando y terminando por un cruce por cero. Necesario para definir las respuestas a tren de ondas, tal como se describen en la sección 3.4.6.

3.4. Especificaciones de funcionamiento

Esta sección describe las características técnicas, incluidas las tolerancias admitidas para las dos categorías, que debe cumplir un sonómetro que cumpla esta Norma. Se entiende

[1] Por ejemplo, para la evaluación de la exposición a ruido en el trabajo según la norma IEC 61652.

que un sonómetro de clase 1 tiene que cumplir todas las especificaciones de dicha Clase. Un sonómetro de clase 2, sin embargo, puede incorporar algunas capacidades de la clase 1.

Según el tipo de instrumento (ver sección 3.2), se establecen unos mínimos de prestaciones: Así, los sonómetros convencionales deben por lo menos indicar niveles de sonido con ponderación frecuencial A y ponderación temporal F. Los sonómetros integradores-promediadores deben poder medir niveles de sonido con ponderación A promediados temporalmente. Por su parte, los sonómetros integradores deben poder indicar niveles de exposición sonora con ponderación A.

En cualquier caso, *todos los sonómetros deben tener la ponderación frecuencial A*, y además los de clase 1 han de tener la ponderación frecuencial C.

El manual de instrucciones del sonómetro debe indicar bajo qué condiciones o configuración se cumplen las especificaciones de funcionamiento, así como el nivel de presión, rango de niveles y dirección de referencia. También la impedancia eléctrica de entrada y los valores de nivel de presión acústica y tensión pico-a-pico más altos que puede soportar el sistema sin sufrir daños.

Las especificaciones de funcionamiento se aplican bajo las condiciones ambientales de referencia, que son las siguientes:

- Temperatura del aire: 23 °C
- Presión estática: 101 325 kPa
- Humedad relativa: 50 %

Los valores de tolerancia dados en la norma incluyen tanto las tolerancias de diseño y fabricación como la incertidumbre de medida máxima permitida en los ensayos de conformidad.

3.4.1. Ajuste a los niveles indicados

Una de las especificaciones más importantes de esta sección se refiere a la *calibración*. Se debe especificar un modelo de calibrador acústico según la norma IEC 60942. Un calibrador adecuado debe mantener un nivel estable en un rango amplio de condiciones ambientales. Se prefiere un nivel de calibración de 94 dB SPL (igual a 1 Pa), aunque pueden utilizarse otras. El procedimiento de calibración debe indicarse en el manual de instrucciones.

En el caso de instrumentos autocontenidos que incorporen un micrófono, se deben proporcionar en función de la frecuencia y para la dirección de referencia las correcciones

necesarias para compensar

- La respuesta en frecuencia no plana del micrófono.
- El efecto de las reflexiones en el cuerpo del sonómetro y de la difracción alrededor de éste.
- El efecto de los accesorios, como por ejemplo pantallas antiviento.

3.4.2. Respuesta direccional

El objetivo de diseño con respecto a la respuesta direccional es una respuesta igual a los sonidos procedentes de todas las direcciones. La norma proporciona las tolerancias de respuesta direccional como *diferencia absoluta máxima entre dos niveles presentados para dos ángulos cualesquiera separados $\pm\theta$ grados*. Así, por ejemplo, un sonómetro de clase 1 puede presentar como máximo una diferencia de 1.3 dB entre dos niveles cuya incidencia difiera 30° y 1.8 dB entre dos niveles cuya incidencia difiera 90°, para las frecuencias entre 250 y 1000 Hz.

3.4.3. Ponderaciones frecuenciales

Para las ponderaciones frecuenciales, el objetivo de diseño es 0 dB (es decir, la ponderación no tiene efecto) para 1 kHz, con tolerancias de ±1.1 dB para la clase 1 y ±1.4 dB para la clase 2.

Para las frecuencias nominales de tercio de octava de 10 a 20000 Hz, las ponderaciones A, C y Z[2] se proporcionan en forma de tabla, indicando el valor objetivo y la tolerancia respectiva para cada clase.

La propia norma indica cómo se calculan las ponderaciones, en forma de especificaciones de filtros. Así, la ponderación C se realiza por dos polos de baja frecuencia, dos polos de alta frecuencia y dos ceros a 0 Hz. La ponderación A se realiza añadiendo a la ponderación C en serie dos filtros paso alto de primer orden.

Así, la norma proporciona directamente el módulo de la respuesta en frecuencia de la red de ponderación C:

$$C(f) = 20 \log \left[\frac{f_4^2 f^2}{(f^2 + f_1^2)(f^2 + f_4^2)} \right] - C_{1000} \tag{3.4}$$

[2]La ponderación Z es el caso trivial de que no se aplica ponderación. Por tanto es 0 dB para cualquier frecuencia. Se aplican sin embargo las mismas tolerancias que a las ponderaciones A y C.

Y de forma similar para la ponderación A:

$$C(f) = 20\log\left[\frac{f_4^2 f^4}{(f^2+f_1^2)(f^2+f_2^2)^{1/2}(f^2+f_3^2)^{1/2}(f^2+f_4^2)}\right] - A_{1000} \qquad (3.5)$$

Las frecuencias de los polos deben calcularse a partir de las especificaciones de los filtros proporcionadas en la norma, y los términos de ajuste para conseguir el objetivo de 0 dB a 1 kHz. Se dan los siguientes valores orientativos: f_1 =20.60 Hz, f_2 =107.7 Hz, f_3 =737.9 Hz, f_4 =12 194 Hz, C_{1000} =−0.062 dB, y A_{1000} =−2.000 dB.

3.4.4. Ponderaciones temporales

Las ponderaciones temporales definidas, en función de su constante de tiempo τ, son la F (*Fast*, rápida), de τ = 0.125 s, y la S (*Slow*, lenta), de τ = 1 s. La norma además especifica que los tiempos de caída para una señal eléctrica de entrada sinusoidal de 4 kHz deben ser de por lo menos 25 dB/s para la ponderación F y entre 3.4 dB/s y 5.3 dB/s para la S. Asimismo, para una señal eléctrica sinusoidal continua de 1 kHz al nivel de presión acústica de referencia, la desviación entre la indicación de nivel medido con ponderación A y ponderación temporal F, nivel medido con ponderación A y ponderación temporal S y nivel con ponderación A promediado en el tiempo no debe exceder ±0.3 dB.

3.4.5. Linealidad de nivel y ruido intrínseco

El sistema debe responder de forma lineal. En concreto, cualquier cambio de la señal de entrada entre 1 y 10 dB debe producir el mismo cambio en el nivel indicado. Para cualquier frecuencia, se considera un *rango de funcionamiento lineal* que debe indicarse en el manual de instrucciones; además, el rango no debe ser menor de 60 dB a 1 kHz. El error de linealidad de nivel, medido como desviación con respecto a una respuesta esperada, no debe exceder ±1.1 dB para la clase 1 y ±1.4 dB para la clase 2.

Debe asimismo indicarse en el manual el nivel más alto de ruido intrínseco esperado para cada modelo de micrófono utilizable en el sonómetro, y para la entrada de señal eléctrica si la hubiera, además del procedimiento a seguir en el caso de que se midan sonidos de nivel bajo para los que el nivel de ruido intrínseco no sea despreciable.

3.4.6. Respuesta a un tren de ondas

El motivo de especificar respuestas a trenes de ondas (definidos en la sección 3.3) es el de cuantificar la respuesta transitoria esperada del sonómetro. La norma define unas diferencias objetivo de diseño entre el nivel indicado por el sonómetro ante trenes de onda de distinta duración y el nivel de sonido de la señal continua, con las tolerancias correspondientes a cada clase.

Para los sonómetros convencionales, la respuesta de referencia es

$$\delta_{ref} = 10 \log \left(1 - e^{-T_b/\tau}\right) \tag{3.6}$$

Donde T_b es la duración del tren de ondas en segundos, y τ es la constante de tiempo de la promediación temporal.

Para los sonómetros integradores e integradores-promediadores, la respuesta se aproxima como

$$\delta_{ref} = 10 \log \left(T_b/T_0\right) \tag{3.7}$$

Donde T_0 es la duración de referencia, igual a 1 s.

3.4.7. Indicaciones de rango

El sonómetro debe estar dotado de un indicador de sobrecarga. Este indicador se activará cuando se alcance un nivel que supere el rango de funcionamiento lineal, en señales continuas, o cuando la respuesta a trenes de onda se salga de la tolerancia indicada. Este indicador deberá permanecer activado hasta que se reinicien los resultados de medición.

También se debe indicar si la medición es menor que el límite inferior de un rango de funcionamiento lineal, condición que la norma denomina "por debajo del rango". Esta indicación debe mantenerse mientras dure la condición o durante un segundo, lo que sea mayor.

3.4.8. Nivel C de pico

Los sonómetros pueden medir niveles de pico con ponderación C, L_{Cpk}. El rango de niveles de pico que debe poder medirse dentro de los límites de tolerancia debe ser de al menos 40 dB. Las especificaciones se indican como diferencia entre niveles de pico correspondientes a ciclos o semiciclos de señales sinusoidales y niveles continuos L_C correspondientes a

una señal sinusoidal continua de la misma frecuencia. Así por ejemplo, para un sonómetro de clase 1, $L_C - L_{Cpk} = 3.5 \pm 2.4\,\mathrm{dB}$.

3.4.9. Presentación de resultados y salida analógica

El dispositivo de presentación de resultados debe indicar de forma clara la magnitud o magnitudes que se están midiendo, con una resolución de 0.1 dB en un rango de niveles de al menos 60dB, mostrando además claramente los datos necesarios para entender la medida, a saber, ponderación frecuencial y temporal, tiempo de promediado, etcétera. Las distintas medidas que se muestran deben estar debidamente explicadas en el manual de instrucciones del sonómetro.

Si el dispositivo de presentación de resultados es digital y se actualiza periódicamente, el valor mostrado debe ser el medido *en el instante de la actualización*. Y si se proporciona una salida digital, se debe describir el método de transferir los datos a un almacenamiento o dispositivo de presentación externo.

3.4.10. Otras especificaciones

La norma especifica otros aspectos del sonómetro tales como la alimentación eléctrica del dispositivo [3], las posibilidades de éste para medir y seleccionar tiempos de medida (necesario para sonómetros integradores e integradores-promediadores), la diafonía en sonómetros multicanal y las emisiones de radiofrecuencia del aparato.

3.5. Criterios ambientales, electrostáticos y de radiofrecuencia

Un sonómetro es un dispositivo de medición que debe ser capaz de funcionar correctamente en un amplio rango de situaciones. Por ello, la norma define unas tolerancias para cada clase en las que debe acotarse la desviación entre la medida a las condiciones de referencia (ver sección 3.4) y la medida a las condiciones especificaciones. En la tabla 3.1 se resumen las tolerancias con respecto a la temperatura ambiente, la presión estática y la humedad relativa. Además, un sonómetro debe ser capaz de seguir funcionando después

[3]Por ejemplo, si el dispositivo usa baterías, se debe indicar cuándo éstas necesitan reemplazarse, y si se utiliza alimentación de red, las tensiones de alimentación máxima y mínima. Además la indicación de

	Presión estática		Temperatura	Humedad relativa
	65–85 kPa	85–105 kPa		25 %–90 %
Clase 1	±1.2 dB	±0.7 dB	−10 °C–50 °C: ±0.8 dB	±0.8 dB
Clase 2	±1.9 dB	±1.0 dB	0 °C–35 °C: ±1.3 dB	±1.3 dB

Tabla 3.1: Resumen de las condiciones ambientales

de una descarga electrostática de ±8 kV, sin perder la configuración o datos almacenados ni quedar dañado. En cuanto a las interferencias electromagnéticas, el funcionamiento del sonómetro no debe verse alterado por campos electromagnéticos a la frecuencia de la red de alimentación eléctrica.

3.6. Dispositivos auxiliares

Un sonómetro puede incluir diversos dispositivos auxiliares, tales como cables de extensión para conectar entre el micrófono y el resto de componentes del micrófono, o dispositivos antiviento o de protección climática. El efecto de cualquiera de estos dispositivos auxiliares sobre las medidas debe estar debidamente recogido en forma de tablas de efecto del accesorio sobre las características acústicas relevantes, tales como sensibilidad, respuesta direccional o ponderación frecuencial. Debe especificarse asimismo si el sonómetro sigue cumpliendo las especificaciones de su clase con el accesorio instalado.

La norma también menciona que un sonómetro puede incluir filtros paso banda internos o externos para el análisis espectral, aunque cómo se realiza dicho análisis no es objetivo de esta norma. Solamente se indica que se debe describir cómo realizar mediciones de nivel de sonido filtrado.

3.7. Manual de instrucciones

El manual de instrucciones de un sonómetro conforme con la norma debe incluir todo lo que se indica en ella. Entre otras cosas, se deben indicar todas las especificaciones electroacústicas indicadas en la sección 3.4, las especificaciones de funcionamiento con respecto a las condiciones ambientales (sección 3.5), y las instrucciones respecto a los dispositi-

nivel no debe diferir más de ±0.3 dB (para clase 1) entre estas dos tensiones.

vos auxiliares (sección 3.6). Además debe incluirse cierta información de funcionamiento e información relativa a los ensayos de evaluación, como se indica en las siguientes secciones.

3.7.1. Información de funcionamiento

Generalidades Descripción del tipo de sonómetro y de la clase de funcionamiento que cumple. Descripción de las partes del sonómetro y de su configuración, indicaciones de cómo configurarlo para poner en marcha una medida, así como instrucciones para la instalación de dispositivos auxiliares si procede.

Aspectos de diseño Descripción de todas las magnitudes acústicas que el sonómetro es capaz de medir, de las ponderaciones frecuenciales y temporales proporcionadas y de los controles de rango de niveles. Además, descripción de todos los todos los dispositivos de presentación de resultados y la frecuencia de actualización de resultados, si procede.

Alimentación eléctrica Si el sonómetro está alimentado por pilas, recomendación del tipo de pilas aceptables y duración de éstas en funcionamiento continuo, y método de comprobación de que la alimentación es suficiente para el correcto funcionamiento de la norma. Si el instrumento funciona con conexión a la red, especificación de la tensión y frecuencia nominales adecuadas y de los rango de tolerancia aceptables.

Ajuste a niveles indicados Identificación de los modelos de calibrador acústico adecuados para mantener la medición correcta, así como el método para realizar dicha calibración. Además deberán proporcionarse las tablas de corrección según lo especificado en la sección 3.4.1.

Manejo del sonómetro Procedimiento para medir sonidos principalmente desde la dirección de referencia (que también debe indicarse), minimizando la influencia de la carcasa del instrumento y del operador. Recomendaciones para seleccionar el rango de niveles óptimos, y para tener en cuenta la influencia del ruido intrínseco en la medida de sonidos de nivel bajo. Tiempo necesario para obtener una lectura. Funcionamiento de los indicadores de rango (sección 3.4.7). Procedimiento para la descarga de datos digitales o conexión a otros dispositivos, si procede. Niveles de sonido correspondientes al *nivel más alto* de ruido intrínseco.

3.7.2. Información para los ensayos

Con el fin de que un laboratorio externo pueda evaluar la conformidad del instrumento con lo especificado en esta norma, se deben indicar diversas informaciones, entre otras:

- Nivel y rango de niveles de presión acústica de referencia, y punto de referencia del micrófono.
- Datos de ajuste para obtener niveles de sonido ponderados A equivalentes a la respuesta a ondas planas sinusoidales, cuando se utiliza un calibrador acústico o actuador electrostático.
- Rango de funcionamiento lineal, expresado como una tabla de niveles ponderados A superior inferior a las frecuencias de 31.5 Hz, 1 kHz, 4 kHz y 8 kHz. Para los sonómetros de clase 1, además deberá especificarse a 12.5 kHz.
- Niveles de sonido ponderados temporalmente y/o promediados en tiempo correspondientes al nivel de ruido intrínseco más alto, indicados para el rango de niveles más sensible.

3.8. Anexos

La norma introduce dos anexos normativos y dos informativos.

Anexo A (normativo) Introduce las incertidumbres expandidas de medida asociadas a las tolerancias proporcionadas en la norma. En el anexo A de este Proyecto se comenta el concepto de incertidumbre expandida de medida y cómo se aplica a esta Norma.

Anexo B (informativo) Comenta brevemente la ponderación AU para la medición de sonido en presencia de ultrasonidos, según lo especificado en la norma IEC 61012.

Anexo C (informativo) Especifica la ponderación temporal I (impulso). Esta ponderación, pensada para la valoración de sonidos impulsivos, ha caído en desuso puesto que los organismos internacionales (como el Comité Técnico 43 de la ISO) encontraron que no es adecuada para la valoración de la sonoridad de estos sonidos, ni para evaluar el riesgo de daño auditivo.

La ponderación I es una ponderación temporal exponencial, de $\tau = 35\,\text{ms}$, a la que se añade un detector de señal que mantiene el nivel medido durante el tiempo suficiente para que se muestre en el dispositivo de presentación de resultados.

Anexo ZA (normativo) Es un anexo a la bibliografía, en el que se indican otras normas internacionales citadas.

Parte II

Desarrollo del Proyecto

Capítulo 4

Desarrollo de la aplicación

SonoPhone es la aplicación iOS que forma la parte *software* de este Proyecto. A pesar de contar con una UI minimalista, se trata de una app completamente funcional en su tratamiento del audio que implementa el procesado necesario para la medida del nivel de presión sonora.

4.1. Estructura y organización

SonoPhone, como casi todas las aplicaciones iOS, se estructura siguiendo el patrón MVC, como se explicó en la sección B.3. Por tanto, el código de las clases se separa en archivos `.m` y `.h` separados por su rol dentro de ese patrón. Las clases tienen el prefijo común `Sono`. Además, existen una serie de *Clases auxiliares*. El árbol de archivos de SonoPhone se muestra en la figura 4.2. Los roles cumplidos por cada clase, así como la relación entre ellas, están esquematizados en la figura 4.1. El Modelo consta de las clases `SonoModel` y `SonoMeasurement`.

La Vista está formada por un *storyboard* (ver sección B.5), que contiene las dos vistas de la aplicación, `SonoView` y `SonoCalibrationView`, y la transición entre ellas.

El Controlador principal es el controlador de la vista `SonoView`, ya que al ser una aplicación con una UI simple no es necesario implementar un "controlador de controladores". Existe además un controlador de navegación que coordina la transición entre las vistas del *storyboard*. A continuación se hará un recorrido por las clases más importantes de la aplicación.

4.2. Modelo: la clase `SonoModel`

Esta es la clase que contiene la interacción con el audio. Maneja la Audio Queue, procesa las muestras, calcula los niveles de acuerdo con la ponderación que se seleccione y gestiona las medidas.

4.2.1. Interfaz

La interfaz, parte pública de la clase, contenida en el fichero `SonoModel.h`, contiene los siguientes elementos:

Constantes Como por ejemplo la duración del *buffer* de audio en segundos, o el número de *buffers* que tendrá la AQ.

Especificación de los protocolos de los delegados Permite a otras clases saber los métodos que necesitan implementar para convertirse en delegados de la clase. Estas especificaciones se detallan en las secciones 4.2.3 y 4.2.4.

Propiedades públicas Se clasifican informalmente según lo que representan.

- Estado: indican en qué estado se encuentra el modelo. Hay dos propiedades booleanas que expresan el estado: `isRunning` se activa cuando la AQ se activa y empieza a recibir datos, y `isMeasuring` cuando estos datos se están procesando y grabando.
- Datos: contiene solo una propiedad, `SPL`, que almacena el nivel de sonido instantáneo. Esta propiedad se va actualizando cada vez que se lee un *buffer* nuevo.
- Metadatos: `SonoFreqWeighting` es una variable de tipo enumeración, que puede tomar valores correspondientes a las ponderaciones frecuenciales A, C y Z.
- Delegados: `delegate` y `calibrationDelegate` son dos objetos de tipo anónimo (`id`), que cumplen los protocolos correspondientes. Así, un objeto que se registre como delegado o delegado de calibración de `SonoModel` recibirá notificaciones relativas al funcionamiento de SonoModel.

Prototipos de métodos Los métodos públicos de SonoModel son:

- `+(id)sharedSonoModel`: Constructor de la clase. SonoModel sigue el patrón *singleton*, es decir que sólo puede existir una instancia de la clase a la vez. Lla-

mando a este método de clase, los otros objetos obtienen la instancia compartida (*shared*).

- `-(void)startInput`: Inicializa y pone en marcha la AQ. Cada vez que llega un *buffer*, en el *callback*, se obtiene el SPL.
- `-(void)stopInput`: Desactiva y libera los recursos de la AQ.
- `-(void)startMeasurement` Comienza una nueva medida, creando una instancia de la clase SonoMeasurement y configurando el callback para que procese las muestras, almacenando el resultado de dicho procesamiento en memoria.
- `-(void)stopMeasurement` Detiene la medida y ordena a la instancia de SonoMeasurement que grabe los resultados en el sistema de archivos del teléfono y los envíe a la nube, si procede.
- `-(void)calibrate` Efectúa una calibración sencilla. Suponiendo que se haya expuesto el teléfono a una fuente de ondas sinusoidal de 1 kHz y 1 Pa, mide el nivel durante un intervalo de tiempo, y almacena ese nivel como valor de calibración.

4.2.2. Implementación

La clase SonoModel es la más extensa de toda la aplicación, y se encarga de la funcionalidad principal, a saber, configurar la entrada de audio, procesar las muestras para calcular las medidas y controlar el guardado y transmisión de éstas.

Entre otros detalles, contiene:

Estructuras para manejar el estado Como se describe en la sección B.6.1, es necesario definir una estructura para pasar al callback todo el estado que queramos mantener. De eso se encarga la estructura apropiadamente bautizada como `SonoModelState`. En la estructura, descrita en 4.1, se almacenan entre otros el formato de audio, la AQ y sus *buffers*, *flags* que indican el estado en el que se encuentra el modelo, el SPL actual, la ponderación frecuencial seleccionada, los *buffers* para calcular las ponderaciones (ver sección 4.2.2.1) y los diccionarios que contendrán los valores de la medida. Todos estos son variables que reflejan el estado del modelo en cada instante.

Callback AQ Esta función C, llamada `InputBufferHandler`, que se ejecuta cada vez que la AQ trae un nuevo filtro, tiene una declaración bien definida, como se lee

Listado 4.1: La estructura SonoModelState

```
typedef struct
{
    AudioStreamBasicDescription mFormat;
    AudioQueueRef mQueue;
    AudioQueueBufferRef mBuffers[Sono_NumberOfBuffers];
    UInt32 totalBytes;
    bool isRunning;
    bool isMeasuring;
    bool isCalibrating;
    Float32 SPLevel;
    enum SonoFreqWeighting freqWeighting;
    FilterStateBuffers timeWeightingFilterBuffers;
    FilterStateBuffers
     freqWeightingFilterBuffers[3];
    CFMutableDictionaryRef AW_AVGBuf;
    CFMutableDictionaryRef timeW_Buf;
    AverageBuffer calibrationBuffer;
    float calibrationValue;
    CFDateFormatterRef formatter;
} SonoModelState;
```

Listado 4.2: InputBufferHandler

```
static void InputBufferHandler(
   void *                                inUserData,
   AudioQueueRef                         inAQ,
   AudioQueueBufferRef                   inBuffer,
   const AudioTimeStamp *                inStartTime,
   UInt32                                inNumPackets,
   const AudioStreamPacketDescription*   inPacketDesc)
```

Listado 4.3: Extracto de la implementación de InputBufferHandler

```
SonoModelState * myState = (SonoModelState *)inUserData;
// ...
AudioQueueLevelMeterState *meters;
UInt32 sizeLvlMtr = sizeof(AudioQueueLevelMeterState)
    * myState->mFormat.mChannelsPerFrame;
meters = malloc(sizeLvlMtr);
error = AudioQueueGetProperty(
    inAQ,
    kAudioQueueProperty_CurrentLevelMeterDB,
    meters,
    &sizeLvlMtr);
if (error) NSLog(@"Error getting level");
else
{
    myState->SPLevel = meters[0].mAveragePower;
    // ...
}
free(meters)
```

en el listado 4.2. La estructura `SonoModelState` llega como puntero anónimo en `inUserData`. El primer paso por tanto es hacer un *casting* al tipo correcto para poder acceder a sus datos. El audio llega en forma de otro puntero anónimo dentro del `inBuffer`, y se conoce su formato por medio del que está almacenado en la estructura `SonoModelState`, y la cantidad de datos que vienen por el parámetro `inNumPackets`. Para extraer el SPL del audio que llega y mostrarlo, se aprovecha la propiedad de la AQ llamada `CurrentLevelMeterDB`, que toma la forma de un array de n estructuras, una por cada canal de entrada, que contienen el nivel medio y el nivel de pico. El nivel medio se almacena en la variable `SPLevel` de la estructura SonoModelState, de donde el accesor lo extraerá a la propiedad SPL (ver sección correspondiente).

Una vez terminado todo el tratamiento que se haya de hacer con el audio que trae el *buffer*, hay que volver a poner este en la cola, llamando a la función `AudioQueue-EnqueueBuffer`.

Funciones de procesamiento Dentro del callback, se comprueba si el estado es de medi-

ción (el flag `isMeasuring` estará activado). En ese caso, se transforman las muestras de audio a formato canónico[1], y se pasan a una función llamada `processSamples`, que realiza los siguientes pasos:

- Conversión a formato de punto flotante, para su procesado posterior[2], y desentrelazado, si las muestras representan audio en estéreo[3].
- Realización del filtrado correspondiente a la ponderación frecuencial seleccionada, como se indica en la sección 4.2.2.1.
- Elevación al cuadrado de las muestras e integración, para la obtención del sonido total promediado en un tiempo igual a la duración del *buffer*. Posteriormente, se usarán todos estos valores en el cálculo del nivel continuo equivalente.
- Paralelamente, se obtiene el nivel instantáneo con ponderación *Fast*, para obtener una historia de los niveles durante todo el tiempo de medida.

Accesores Algunas de las propiedades públicas de SonoModel reflejan el estado interno, almacenado en la estructura. Otras propiedades son privadas, como la instancia de `SonoMeasurement`. Merece especial atención el metodo *getter* de la propiedad `freqWeighting`, en el que se inicializan los coeficientes del filtro, como se verá en la sección 4.2.2.1.

Inicializador y destructor El método `init` inicializa todo lo necesario, como por ejemplo los *buffers* de los filtros. El destructor libera toda la memoria que se asignó manualmente.

Inicio y pausa de AQ El método `startInput` primero llama a una función auxiliar que inicializa el formato (contenido en la estructura `mFormat` dentro de SonoModelState), con los siguientes parámetros:

- Número de canales y frecuencia de muestreo: se interroga a la Audio Session sobre las capacidades del dispositivo para establecer estos parámetros.

[1] Ver sección B.6. El tipo de dato `AudioSampleType` representa este formato. La conversión se puede hacer directamente mediante un *casting* porque el formato de audio no es comprimido.

[2] Esto está condicionado por el framework dedicado al procesado de señal, Accelerate, que requiere este formato. También va a determinar la estructura del filtrado descrito en la sección 4.2.2.1.

[3] Desentrelazar consiste simplemente en tomar una de cada n muestras. Para ello, las funciones de Accelerate tienen un parámetro llamado *stride* que controla cuántas muestras de la señal se toman.

- Formato: PCM lineal, 16 bits por muestra por canal.
- Flags de formato: agrupan diversos parámetros de las muestras, como si se aplica entrelazado, si son de punto flotante o enteras, etcétera. En SonoPhone se aplica la flag `kAudioFormatFlagsCanonical`, que representa el formato canónico[4], es decir *entero, entrelazado, con* endianness *nativa y totalmente ocupado*[5].

Después, `startInput` crea la AQ y llama a otra función auxiliar que crea los *buffers* y los añade a la AQ. Para determinar el tamaño de memoria que es necesario asignar, se calcula a partir de los campos de bits por muestra y muestras por canal de la estructura de formato.

El último paso necesario para poner en marcha la AQ es asignar la propiedad de ésta que permite obtener el SPL, llamada *metering*.

El proceso que sigue `stopInput` para parar la AQ es conceptualmente opuesto al de `stopInput`. En primer lugar interroga a la AQ si está activa, leyendo su propiedad `IsRunning`. Si es el caso, llama a las funciones `AudioQueueStop`, primero, y `AudioQueueDispose`, después, para parar la AQ y liberar los recursos que utiliza.

Inicio y pausa de medida De esta funcionalidad se encargan los métodos `startMeasurement` y `stopMeasurement`. Se trata de obtener una instancia de la clase SonoMeasurement, asignar sus metadatos, tales como fecha y hora de inicio y localización, e inicializar los objetos que van a ir almacenando las medidas. Estos son objetos Core Foundation, de la clase `CFMutableDictionary`; almacenan en un formato clave-valor en el que aquella va a ser el momento exacto en el que se obtiene cada dato y éste el dato en sí. Al parar la medida se asignan los metadatos finales, principalmente la fecha y hora de finalización, y se pide al objeto SonoMeasurement que efectúe los cálculos que sean necesarios y que almacene sus medidas en el sistema de archivos, acción conocida como *persistencia*, o las envíe a la nube.

Calibración Para efectuar la calibración se recurre a un tipo especial de *buffer* llamado `AverageBuffer`. Es un *buffer* circular que almacena las últimas muestras que llegan a él. En el callback, cuando está activada la calibración, se va escribiendo en este *buffer*, y cuando se termina el intervalo de tiempo especificado (ver 4.2.1, apartado "Calibración"), se calcula la media de todos los valores. Para esto se utiliza un

[4] Ver sección B.6.

[5] Totalmente ocupado (*packed*) significa que todos los bits de la muestra son significativos.

mecanismo de Objective-C que permite mandar a un objeto un mensaje que indica que ejecute un selector (ver sección B.1) después de un intervalo de tiempo. El valor de calibración obtenido se almacena en las *preferencias de usuario*, un archivo de metadatos que todas las apps contienen.

4.2.2.1. Ponderación frecuencial y temporal: filtros

La implementación del filtrado necesario para realizar la ponderación frecuencial y temporal es uno de los aspectos principales de SonoPhone.

Para la ponderación frecuencial A o C, se parte de las especificaciones del filtro según la norma (ver sección 3.4.3). A continuación se diseña el filtro digital siguiendo alguno de los métodos de diseño comúnmente utilizados.

Para el diseño del filtro, se ha utilizado el paquete Octave para MATLAB[6]. Este *toolbox* contiene una función llamada `adsgn` que diseña un filtro IIR acorde a las especificaciones de la norma. Esta función calcula directamente el numerador y denominador de la función de transferencia, y usando la transformada bilineal obtiene los coeficientes del filtro digital.

Sin embargo, este filtro, de orden 7, no puede implementarse con facilidad en la aplicación. La razón es que, para obtener el máximo rendimiento, la aplicación utiliza la API `vDSP` del framework Accelerate de iOS, que está optimizada al máximo para la arquitectura de los dispositivos. Esta librería solo proporciona una función que realice el filtrado recursivo, llamada `vDSP_deq22`, y ésta solo permite la implementación de un filtro bicuadrático, es decir, de la siguiente forma:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.1}$$

Por lo tanto, es necesario encontrar una manera de realizar el filtrado completo como una serie de secciones bicuadráticas. Para ello, se usó la función `tf2sos` del *Signal Processing Toolbox*. Esta función convierte una función de transferencia, expresada como los coeficientes de su numerador y denominador, en una matriz $L \times 6$ de coeficientes de cada etapa bicuadrática, o sección de segundo orden (SOS, *Second Order Section*), y un término de ganancia global.

Cada fila de esta matriz tiene la forma $[b_{0k}\, b_{1k}\, b_{2k}\, 1\, a_{1k}\, a_{2k}]$, coeficientes de la k-ésima

[6]No confundir con GNU Octave, programa de computación compatible con MATLAB. Octave es un paquete diseñado por Christophe Couvreur que se puede encontrar en `http://www.mathworks.es/matlabcentral/fileexchange/69-octave`.

Listado 4.4: Coeficientes del filtro para la ponderación A, y su descomposición en SOS

```
[B, A] = adsgn(Fs);
> B = [0.2557 -0.5115 -0.2557  1.0230 -0.2557 -0.5115  0.2557];
> A = [1.0000 -4.0196  6.1894 -4.4532  1.4208 -0.1418  0.0044];
[SOS, G] = tf2sos(B,A);
> SOS = [1.0000 -2.0000  1.0000  1.0000 -1.8849  0.8864;
         1.0000  2.0000  1.0000  1.0000 -1.9941  0.9941;
         1.0000 -2.0000  1.0000  1.0000 -0.1405  0.0049];
> G = 0.2557;
```

Listado 4.5: Estructura que contiene el estado de una etapa del filtro

```
typedef struct
{
    float * gCoefBuffer;
    float * gInputKeepBuffer;
    float * gOutputKeepBuffer;
    float gGainFactor;
} FilterStateBuffers;
```

SOS. En 4.4, se ilustra este proceso. Como se puede ver, se necesitarán tres etapas SOS para la implementación del filtro.

Para implementar cada una de estas etapas del filtro, puesto que la señal se va a filtrar en bloques, hace falta una estructura que mantenga el estado del filtro. Esta estructura (ver 4.5) contendrá:

- Un *buffer* que almacena los 6 coeficientes de cada etapa.
- Dos *buffers* que mantienen el estado de los acumuladores del filtro entre bloque y bloque, uno para la entrada y otro para la salida. Este estado son dos muestras que antes de cada llamada a `vDSP_deq22` se copian a los *buffers* de entrada y salida, respectivamente, garantizando que no haya transiciones bruscas entre los bloques.
- El término de ganancia del filtro. En el caso de que el filtro en cuestión sea una etapa de varias, se almacena aquí el global y solo se aplica el de uno de ellas, por simplicidad.

El procesado lo realiza una función llamada `processWithIOData`, que recibe como pará-

metros los datos, en forma de puntero a `float`, el número de muestras (*frames*) a procesar, y la estructura `FilterStateBuffers` del filtro. El bloque de datos procesado vuelve en el propio parámetro, que es de entrada y salida, de ahí el nombre de `IOData`.

Los pasos que sigue el algoritmo son:

1. Crear una variable llamada `inputBuffer`, de tamaño `frames + 2`.
2. Crear otra variable llamada `outputBuffer`, del mismo tamaño.
3. Copiar el `gInputKeepBuffer` en `inputBuffer`.
4. Copiar el `gOutputKeepBuffer` en `outputBuffer`.
5. Copiar `IOData` en el tercer elemento de `inputBuffer`.
6. Filtrar `inputBuffer` con `vDSP-deq22`. La salida aparece en `outputBuffer`.
7. Copiar `outputBuffer` en `IOData`.
8. Copiar los dos últimos elementos de `inputBuffer` en `gInputKeepBuffer`.
9. Copiar los dos últimos elementos de `outputBuffer` en `gOutputKeepBuffer`.
10. Liberar memoria de `inputBuffer` y `outputBuffer`.

El estado, representado por los *keepBuffers*, no necesita transmitirse entre etapas, sino entre sucesivas iteraciones de procesamiento en la misma etapa. Por ello cada etapa tiene su propia estructura de estado.

4.2.3. Protocolo `SonoModelDelegate`

La clase SonoModel usa este protocolo para transmitir mensajes al objeto delegado cuando ocurre un acontecimiento relevante a la entrada de audio. Contiene los siguientes métodos:

- inputWasStarted: se emite este mensaje cuando `startInput` ha inicializado correctamente la AQ y sus *buffers* correspondientes.
- inputWasStopped: cuando `stopInput` ha parado y liberado los recursos de la AQ.
- measurementWasStarted: cuando `startMeasurement` ha inicializado lo necesario para empezar la medida.

- measurementWasStopped: cuando `stopMeasurement` ha terminado la medida y escrito todos los metadatos.

El controlador de vista, `SonoViewController`, se establece como delegado para actualizar la UI principal en respuesta a estos sucesos.

4.2.4. Protocolo `SonoModelCalibrationDelegate`

Pensado para que el controlador `SonoCalibrationViewController` sepa cuándo empieza y termina la calibración.

- calibrationWasStarted: se emite cuando la calibración comienza.
- calibrationWasFinished: se emite cuando acaba la calibración y se ha almacenado el valor obtenido.

4.3. Modelo: la clase `SonoMeasurement`

SonoMeasurement se encarga de manejar todo lo relativo a una *medida*. En el contexto de SonoPhone, una medida se define como "el conjunto de los datos obtenidos con el procesamiento de las muestras de audio obtenidas de la entrada y todos los metadatos relevantes".

Los *datos* de la medida son:

- Una *serie temporal* de valores de sonido ponderados frecuencialmente, promediados durante la duración del *buffer*, almacenados en una estructura tipo diccionario en la que la *clave* es el instante en el que se obtuvo el valor, y el *valor* en sí es único para cada clave. Esta serie temporal es la base para el cálculo de todos los demás datos.
- Una serie temporal de valores de sonido ponderado frecuencialmente *y temporalmente*, para la posterior detección de picos de sonido.
- El valor del nivel de sonido continuo equivalente, obtenido a partir de la integración de la serie temporal a lo largo de la duración total de la medida.
- Valor de calibración utilizado para convertir las muestras adimensionales a valores físicos de sonido. Toma la forma de un valor en decibelios que debe sumarse al nivel

respecto a fondo de escala para obtener el SPL[7].

- Opcionalmente, otros datos de interés calculables a partir de la serie temporal, como nivel máximo y mínimo alcanzado o niveles percentiles[8]. En SonoPhone no se han incorporado estos cálculos en la primera versión, aunque podrían añadirse en versiones futuras.

Los *metadatos* que contiene la medida pueden ser, entre otros:

- Tiempo de inicio y de fin: Con exactitud de milisegundo, permiten dar una referencia temporal a los datos y determinar la duración de la medida, fundamental para el cálculo del nivel continuo equivalente y otras magnitudes integradas-promediadas.

- Localización: El framework Core Location de iOS proporciona una API que permite conocer la localización física del dispositivo con diversos grados de exactitud, desde pocos metros (usando el GPS) a varios kilómetros, si se usa la triangulación con antenas móviles. Esto permite añadir a los metadatos la latitud, longitud y altitud del punto en el que se mide. Esto es fundamental si se quiere usar SonoPhone para la elaboración de mapas de ruido.

- Ponderación frecuencial y temporal utilizadas, así como la duración del *buffer*.

- Identificador del usuario: Para la elaboración de informes de ruido, es fundamental identificar al operador del sistema de medida. La manera de identificar al usuario puede ser mediante un apodo que él o ella elija, o de forma automática, si el sistema le asigna una ID única. En SonoPhone de momento no se ha implementado esta funcionalidad.

- Breve descripción: De forma similar, es útil añadir un pequeño texto que permita distinguir la medida de otras.

- Otros: Si se ha utilizado un micrófono externo para realizar la medición, sería conveniente indicar el modelo o incluso el número de serie de éste. Las condiciones

[7]El nivel respecto a fondo de escala, conocido como dBFS, se calcula como veinte veces el logaritmo decimal del valor absoluto de la muestra dividido por el valor máximo representable en el sistema. Es siempre negativo por definición.

[8]Se conoce como *nivel del percentil N*, y se representa como L_N, el nivel de sonido que se ha superado durante el N % de la duración de la medida. Habitualmente se proporcionan valores como L_{90} o L_{10}.

meteorológicas también son un factor relevante para muchas medidas de ruido. En general cualquier metadato que se añada debe aportar información sobre el *contexto* en el que se realiza la medida.

SonoMeasurement expone en su interfaz pública sus *diccionarios*, en donde almacena los datos. Esto le permite actuar como subsidiaria de otro objeto, en este caso SonoModel, que efectúa el "trabajo pesado" de obtener los datos y los almacena en una instancia de SonoModel que posea.

Las propiedades públicas de SonoMeasurement se dividen entre las que se pueden asignar por otro objeto, como la descripción, o los mismos diccionarios de datos "en crudo", y las que SonoMeasurement calcula internamente, como la localización, el valor del nivel continuo equivalente o el nivel de pico.

El método más importante de la clase es `persistMeasurement`, que realiza el proceso de compilar todas los datos y metadatos en un objeto y *serializar* éste. Serializar significa convertir los datos almacenados en la memoria en un formato que permita su almacenamiento y transmisión, permitiendo además la recreación de estos datos en otro momento y lugar. Cocoa proporciona APIs para convertir sus objetos a XML[9], aunque se ha elegido JSON como formato para la serialización debido a que es un formato abierto, compacto y ampliamente utilizado, sobre todo en la web[10].

Una vez serializado, el archivo se almacena en el teléfono, dentro de la carpeta *Documents* de la aplicación, en forma de fichero de texto con extensión `.json`, y se sube a la nube, efectuando la petición PUT al servicio Amazon S3[11].

4.4. Vista principal: `SonoViewController`

El punto de entrada del usuario a SonoPhone es la vista SonoView (ver figura 4.3), que contiene:

Una barra de título Que refleja en cada momento la vista en la que se encuentra el usuario, permitiendo la vuelta a la principal. Esta es una funcionalidad de un controlador

[9] En Cocoa es muy importante el concepto de *property list*, que comprende los objetos básicos que representan valores, como cadenas, números y fechas, las colecciones como arrays y diccionarios, y cualquier combinación de estos. La serialización y deserialización de las *property lists* es fundamental para guardar y recrear el estado del programa, preferencias, etc. cuando sea necesario.

[10] Otra alternativa podría haber sido CSV (valores separados por comas).

[11] Para detalles sobre el almacenamiento en la nube, leer el apéndice E.

de UIKit llamado `NavigationViewController`.

Un interruptor de entrada Al activarse, pone en marcha la entrada de audio. Al desactivarse, la para.

Un selector de ponderación frecuencial Permite cambiar entre las ponderaciones A, C o Z.

Un botón de calibración Redirige al usuario a la pantalla de calibración.

Un botón de comienzo y parada de la medición Pone en marcha una nueva medición o para la ya iniciada. Este botón solo está activo cuando se ha activado la entrada con el interruptor correspondiente.

Un visualizador del SPL Una vista personalizada, llamada `SonoLevelMeter`, recoge el SPL del modelo y lo representa como una barra móvil. Se trata en más detalle en la sección 4.6.

SonoViewController alberga importante funcionalidad. Por ejemplo, en su método `viewDidLoad`, que se ejecuta nada más cargarse la vista, se inicializa la Audio Session con sus correspondientes callbacks para manejar las interrupciones (ver sección B.6.2).

Cuando se pone en marcha la entrada de audio, se activa un temporizador repetido que cada 125 ms va a obtener el SPL del modelo y lo va a mostrar mediante la clase SonoLevelMeter.

SonoLevelMeter es el delegado de SonoModel, de modo que recibe las notificaciones especificadas en la sección4.2.3 y puede actualizar el estado de los controles. Por ejemplo, no se activa el botón de comienzo de medida hasta que no se recibe notificación de que la entrada ha empezado.

4.5. Calibración: `SonoCalibrationViewController`

La pantalla de calibración de SonoPhone es sencilla y autoexplicativa, como se representa en la figura 4.4. Una vez conectada la fuente de sonido tal como se indica, se le da al botón, y se activará la función de calibración del modelo. Como esta clase es delegada de calibración de SonoModel, recibe notificación cuando acaba este proceso, momento el que la vista vuelve automáticamente a la pantalla principal.

4.6. SonoLevelMeter

En iOS, existen tres tipos de vistas, todas subclases de `UIView`:

- Vistas con elementos de UIKit, tales como botones, cajas de texto, etiquetas, interruptores, etcétera.
- Vistas personalizadas. Estas sobreescriben el método `drawRect` de la clase padre para dibujar –mediante una API conocida como Quartz– su representación.
- Vistas OpenGL. Para juegos o aplicaciones de alto rendimiento gráfico, se proporciona este tipo de vista, una pizarra blanca preparada para ser dibujada con este popular entorno gráfico.

Por supuesto casi ninguna de estas existe en aislamiento, sino que son comunes las vistas que combinan varias subvistas de distintos estilos. La vista de calibración es un ejemplo del primer tipo: todos los elementos son estándar. Sin embargo, SoundLevelMeter, que debe modificar su tamaño en función del SPL que llegue, es un ejemplo de la segunda.

Existe un protocolo, `SonoLevelMeterDataSource`, que especifica un método que trae a SonoLevelMeter el nivel que tiene que mostrar en cada momento, es decir entre 0 (no se dibuja el rectángulo) a 1 (se dibuja todo el rectángulo). SonoViewController implementa dicho protocolo y posee la instancia de SonoLevelMeter.

4.7. Configuración de la app

Para configurar la app, se ha de editar un fichero llamado `info.plist`, que contiene pares clave-valor que cambian propiedades como el nombre de la aplicación, los modos de rotación del dispositivo que acepta (vertical, apaisado izquierdo y derecho...).

Una propiedad resulta especialmente interesante: la que activa la compartición de archivos en iTunes. Esta funcionalidad permite acceder cuando se conecta el dispositivo a un ordenador con iTunes, los archivos contenidos en la carpeta Documents de la aplicación. Así se permite recuperar las medidas realizadas.

Otras opciones, establecidas por el usuario, se pueden guardar en los *User Defaults*. Por ejemplo, el valor de calibración obtenido se almacena allí. La dirección web de la TVM para Amazon S3 (ver apéndice E) también es un ejemplo de dato relativo al desarrollo de la aplicación que conviene almacenar en los *User Defaults*.

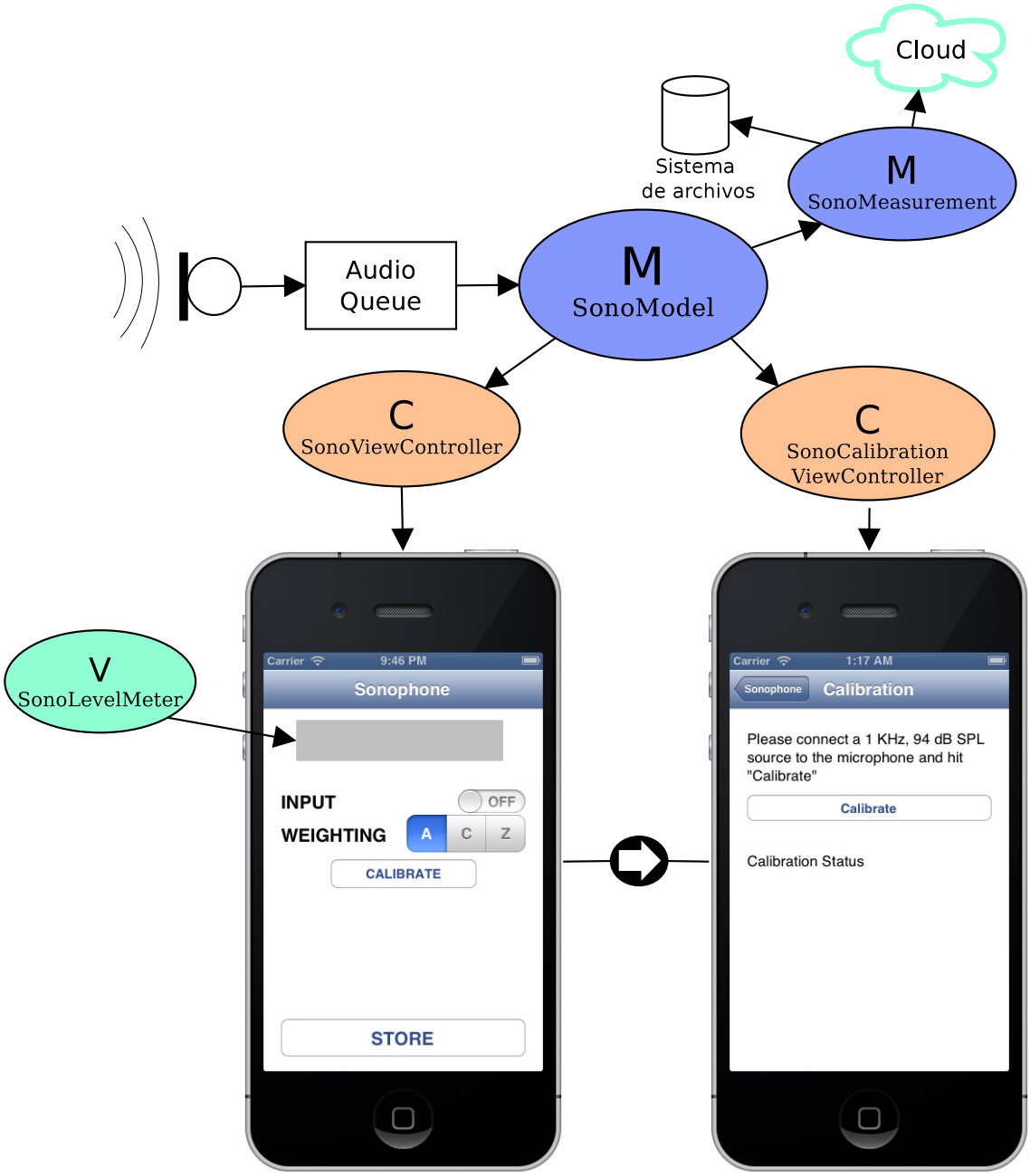

Figura 4.1: Estructura de SonoPhone.

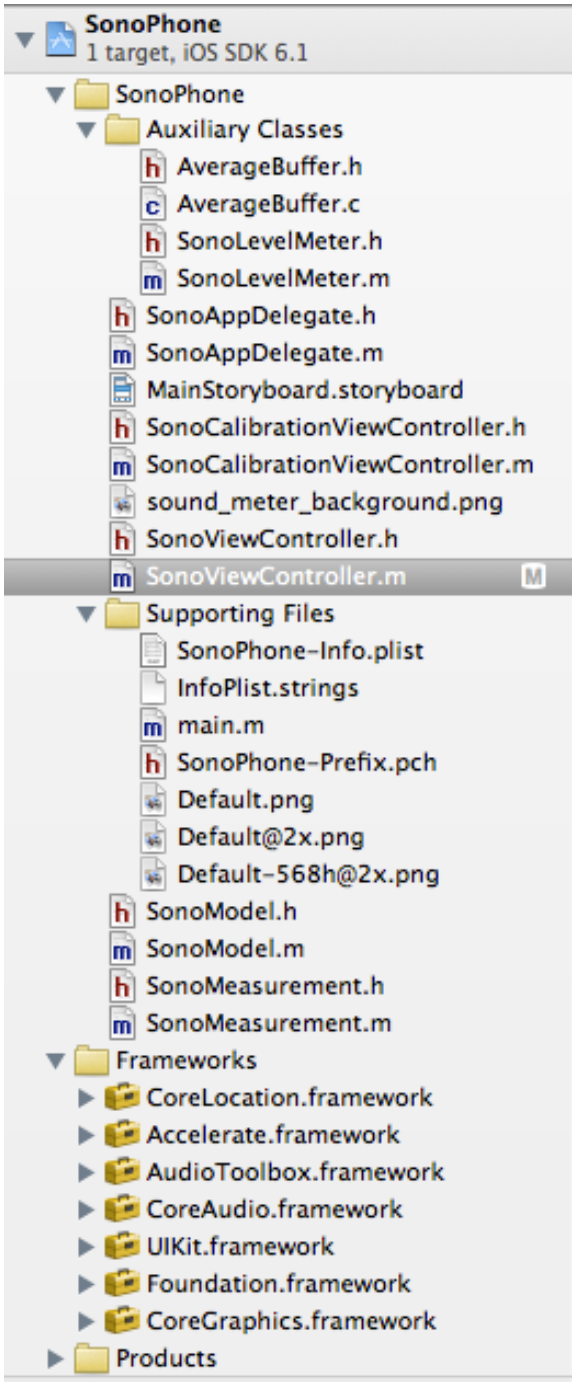

Figura 4.2: Árbol de archivos de SonoPhone, tal como se muestra en XCode.

Figura 4.3: Pantalla de entrada a SonoPhone

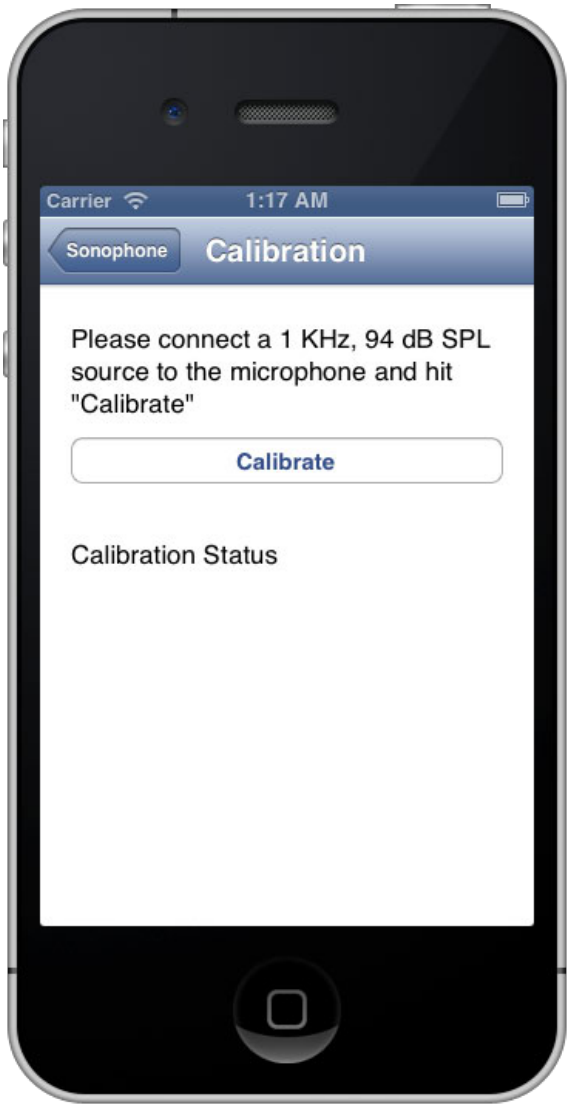

Figura 4.4: Pantalla de calibración de SonoPhone

Capítulo 5

Evaluación del hardware del iPhone

Para cumplir las especificaciones citadas en el capítulo 3, se han de realizar una serie de pruebas de evaluación en laboratorio para determinar en qué grado el sistema se ajusta a la norma. En el caso de SonoPhone, se debe comprobar si el *hardware* del iPhone ofrece las características de precisión, respuesta en frecuencia, etcétera, que permitan implementar una aplicación de prestaciones comparables a los sonómetros existentes en el mercado.

No se pretende realizar una evaluación exhaustiva del sistema para determinar la conformidad del modelo a la Norma, según lo que ésta especifica en su segunda parte (ver Bibliografía), proceso que sería merecedor de un Proyecto en sí mismo, sino tratar de determinar si el *hardware* es válido para el propósito o no.

El audio puede entrar al iPhone por tres vías:

- Micrófono integrado: Pensado para su uso en conversaciones telefónicas, es un micrófono de muy pequeño tamaño tamaño con una respuesta en frecuencia irregular, puesto que está pensado para transmitir claramente la voz. Además, esta entrada contiene filtrado y procesado de señal con el objeto de mejorar la calidad de voz.
- Entrada de audio: Entrada mono analógica, integrada en el mismo conector que la salida estéreo. Diseñada para su uso con sistemas de manos libres, aunque se puede conectar cualquier tipo de micrófono o fuente de audio a esta entrada, mediante el uso de un adaptador[1].
- Conector *dock*: En los modelos anteriores al iPhone 4S, el conector *dock* de 30 pines incluye una entrada de línea. En los modelos posteriores, que usan conector *Lightning*,

[1] Como el iRig, usado para la medida (ver apéndice C).

existe una entrada de audio digital. Diversos accesorios externos se aprovechan de esto para ofrecer entradas de alta calidad. Esta vía no se ha estudiado al no disponer de uno de estos accesorios.

El caso ideal sería que el iPhone pudiera utilizarse como medidor sin la necesidad de ningún micrófono ni accesorio externo. Sin embargo, como se ha indicado, la fiabilidad del micrófono integrado no es muy alta. Se ha elegido medir la respuesta de la entrada de audio por ser la más versátil a la hora de realizar medidas.

5.1. Estudio de la respuesta en frecuencia con ponderación A

La norma IEC 61672 especifica los rangos de valores en los que debe situarse la respuesta en frecuencia ponderada del sistema (ver sección 3.4.3). El objetivo es que la respuesta en tercios de octava se aproxime lo más posible a los valores que la norma especifica en forma tabular, por lo menos para la Clase 2 de funcionamiento, teniendo en cuenta la incertidumbre de medida, según lo indicado en el apéndice A.

En el iPhone, las desviaciones con respecto a la respuesta en frecuencia plana provienen principalmente de:

- Respuesta en frecuencia irregular del micrófono externo usado para la medición, o el micrófono integrado, si se utiliza.
- Respuesta eléctrica de la entrada de audio, hasta el conversor analógico-digital.
- Cualquier procesado digital que sufra la señal antes de llegar a la aplicación: filtrado, cancelación de ruido, etcétera.

Para la evaluación de la respuesta en frecuencia, el primer efecto se anula estimulando el sistema con una señal eléctrica aplicada a la entrada de audio. Todos los sistemas que se encuentran entre la entrada analógica de audio y la salida digital que recoge la aplicación han de tratarse como una "caja negra" que se estudia como un único sistema. La realización de esta medida se detalla en el apéndice C.

Los resultados de la medida, expresados como valores de diferencia, en decibelios, entre el nivel sin ponderar y el nivel ponderado, para las bandas de tercio de octava de 50 a 16000 Hz, se muestran en la tabla 5.1. Para efectuar la comparación con la especificación dada por

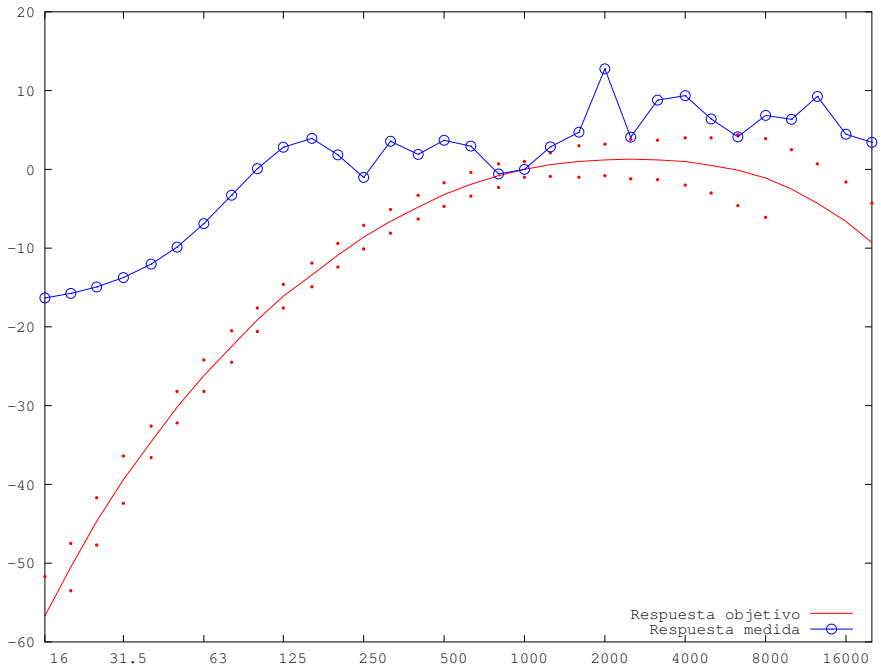

Figura 5.1: Resultados de la medición, comparados con el objetivo de la Norma.

la Norma, se proporcionan también los valores objetivo, así como la tolerancia (entendida ésta como la desviación absoluta máxima admisible) para la clase 2 de funcionamiento.

La figura 5.1 da una representación gráfica de los datos de la tabla 5.1. Para dar significado a la medición, es necesario proporcionar una indicación de la incertidumbre asociada a la medida, según recoge la Norma. Este valor, calculado en la sección C.4, es de 0.5 dB. Esta medición no serviría como ensayo de evaluación de modelo, aunque sí puede ser válida ya que el valor de la incertidumbre es bastante pequeño en relación a los valores que se manejan.

5.2. Interpretación de la medida

El objetivo de la medida es determinar si es posible, utilizando la entrada de audio del iPhone, llegar a cumplir las especificaciones de la norma en lo relativo a la ponderación A.

Se observa en la figura 5.1 que la respuesta del sistema se encuentra en todas las frecuencias por encima del objetivo. La combinación de la respuesta del micrófono y el posterior filtrado que se realice debe conseguir que la respuesta total del sistema se ajuste

Frecuencia central (Hz)	Valor medido (dB)	Objetivo	Tolerancia	Diferencia
16	-16.3	-56.7	$-\infty;+5.0$	40.4
20	-15.7	-50.5	± 3.0	34.7
25	-14.9	-44.7	± 3.0	29.8
31.5	-13.7	-39.4	± 3.0	25.7
40	-12.0	-34.6	± 2.0	22.6
50	-9.9	-30.2	± 2.0	20.3
63	-6.9	-26.2	± 2.0	19.3
80	-3.3	-22.5	± 2.0	19.2
100	0.1	-19.1	± 1.5	19.2
125	2.8	-16.1	± 1.5	18.9
160	3.9	-13.4	± 1.5	17.3
200	1.8	-10.9	± 1.5	12.7
250	-1.0	-8.6	± 1.5	7.6
315	3.5	-6.6	± 1.5	10.2
400	1.9	-4.8	± 1.5	6.7
500	3.7	-3.2	± 1.5	6.9
630	2.9	-1.9	± 1.5	4.8
800	-0.6	-0.8	± 1.5	0.2
1000	0.0	0.0	± 1.0	0.0
1250	2.8	0.6	± 1.5	2.3
1600	4.7	1.0	± 2.0	3.7
2000	12.7	1.2	± 2.0	11.6
2500	4.1	1.3	± 2.5	2.8
3150	8.8	1.2	± 2.5	7.6
4000	9.3	1.0	± 3.0	8.4
5000	6.4	0.5	± 3.5	5.9
6300	4.1	-0.1	± 4.5	4.2
8000	6.8	-1.1	± 5.0	7.9
10000	6.3	-2.5	$-\infty;+5.0$	8.8
12500	9.3	-4.3	$-\infty;+5.0$	13.6
16000	4.5	-6.6	$-\infty;+5.0$	11.1
20000	3.4	-9.3	$-\infty;+5.0$	12.7

Tabla 5.1: Resultados de la medición de respuesta en bandas.

a lo especificado.

La respuesta del sistema resultaría problemática si se encontrara *por debajo* de la especificación, dado que ningún filtrado puede recrear energía que no existe en la señal original. Sin embargo, una desviación positiva como la que existe en el sistema puede compensarse mediante la introducción de un sistema que corrigiera la respuesta del sistema en aquellas bandas donde fuera necesario. En el diagrama de la figura 5.2, se observa que la respues-

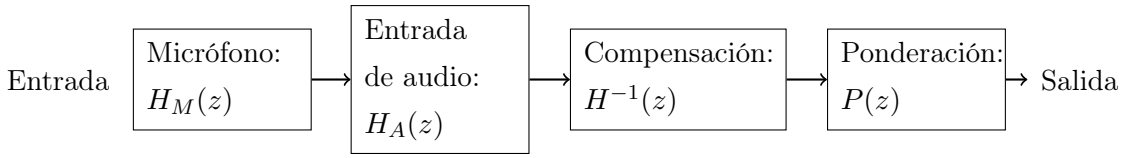

Figura 5.2: Diagrama de bloques de los distintos sistemas que conforman la respuesta total de la entrada.

ta total se puede considerar como formada por varias respuestas de distintos sistemas: el micrófono, la entrada de audio (de la que se ha obtenido la respuesta) y la ponderación $P(z)$, que corresponde al filtro usado en la aplicación, detallado en la sección 4.2.2.1. La respuesta total debe entrar dentro de los límites de tolerancia de la respuesta objetivo dada por la Norma.

Para encontrar $H^{-1}(z)$, se pueden usar distintos métodos. Si se conoce la forma de la respuesta $H_A(z)H_M(z)$ con exactitud, se puede compensar con un filtro inverso, en el que los polos pasan a ser ceros y viceversa. También puede optarse por utilizar técnicas de filtrado adaptativo, en el que los coeficientes del filtro se elijan para minimizar la desviación entre la respuesta deseada y la obtenida.

Una aproximación simple pero posiblemente suficiente para la exactitud requerida sería un sistema tipo *ecualizador gráfico*, es decir, una agrupación en paralelo de tantos filtros paso banda como bandas de tercio de octava se consideren, con ancho de banda igual a cada tercio de octava. Cada filtro puede introducir una atenuación en la banda. Si elegimos la ganancia igual y de signo opuesto a la desviación con respecto a la respuesta especificada en cada banda, lograremos compensar la respuesta desigual del sistema y tener una respuesta adecuada.

En la figura 5.3 se muestran los ajustes de un hipotético ecualizador que compensara la desviación de la respuesta ponderada. Una vez aplicada la corrección, habría que volver

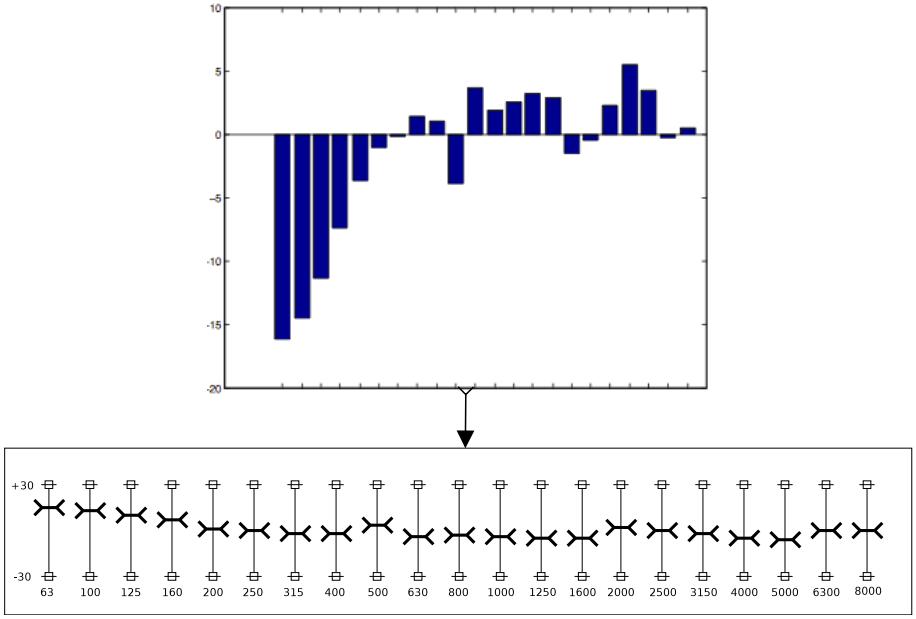

Figura 5.3: Sistema tipo "ecualizador gráfico" aplicado a la entrada para compensar la respuesta no plana de ésta.

a realizar la prueba, ajustando el filtro cuantas veces se necesite hasta que la respuesta se ajuste al objetivo. Es necesario notar que la corrección introduce un nuevo término en la incertidumbre de medida (ver sección C.4).

En conclusión, a la pregunta de si es posible utilizar el iPhone como medidor de presión sonora profesional utilizando la entrada de audio, se puede responder que sí, mediante el diseño cuidadoso de los filtros y realizando las pruebas adecuadas. El ajuste a la norma dependerá en cualquier caso del micrófono que se utilice; sin embargo, si éste presenta una respuesta bastante plana (algo fácil de conseguir con micrófonos de membrana pequeña), se puede conseguir un buen ajuste al objetivo de diseño en un rango amplio de frecuencias, mientras que las propias características de la ponderación A permiten bastantes pérdidas a muy baja y muy alta frecuencia.

Si lo que se pretende es utilizar el micrófono del iPhone, la respuesta es menos clara. En la figura 5.4 se muestran medidas de respuesta en frecuencia realizadas por la empresa Faber Acoustical, fabricante de aplicaciones de medida acústica para iOS, en las que se evidencia un acusado descenso de la respuesta hasta 250 Hz, con una pendiente de unos 24 decibelios por octava. En estas condiciones, la respuesta estaría por debajo del objetivo.

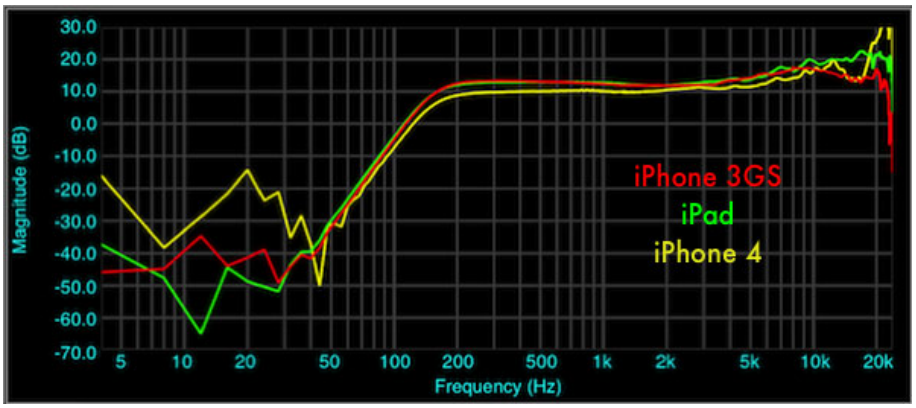

Figura 5.4: Respuesta en frecuencia de diversos dispositivos iOS, obtenida de `http://blog.faberacoustical.com/2010/ios/iphone/iphone-4-audio-and-frequency-response-limitations/` (Consultado en Julio de 2013).

Sin embargo, es posible que esa irregularidad se deba al filtrado paso alto que, antes de iOS 6, el sistema operativo introducía obligatoriamente para evitar ruido de viento y *pops*. Sin embargo, desde esta versión se permite su desactivación mediante la API de Audio Session. Dado que la medida se realizó en 2010, antes de la aparición de iOS 6, es posible que en dispositivos actualizados se pueda cumplir el objetivo.

Parte III

Conclusiones

Capítulo 6

Conclusiones

6.1. Evaluación del cumplimiento de los objetivos

En los objetivos del Proyecto (sección 1.2), se enumeraban los requisitos que SonoPhone debía cumplir para aspirar a ser un sistema de medida correcto.

Precisión: En el capítulo 5 se efectuó una medida de cuánto se ajustan las medidas realizadas con SonoPhone a la especificación de la norma. La conclusión es que, por lo menos usando la entrada de audio, es posible implementar una corrección que proporcione un ajuste correcto a las especificaciones de la Norma para el nivel de presión sonora con ponderación A. El funcionamiento con el micrófono integrado, así como el ajuste a otros parámetros de la Norma, queda como campo de desarrollo futuro.

Fiabilidad: iOS es un sistema operativo muy estable, ampliamente probado y con una gran comunidad de desarrolladores; así, es poco probable que ocurra un error no documentado. Las fuertes restricciones que Apple impone sobre las aplicaciones, revisándolas antes de su distribución y manteniéndolas aisladas unas de otras, hacen que haya una cierta garantía de que la aplicación funcione correctamente y pueda responder a las diversas vicisitudes (memoria o batería baja, ausencia de red, llamadas entrantes), que puedan suceder durante el uso del dispositivo.

Versatilidad: Un iPhone o iPad es una máquina que puede convertirse en muchas máquinas, y esta forma de *tabla rasa* es lo que lo hace atractivo para el desarrollo. De este modo, en los *frameworks* de iOS el desarrollador tiene ya incluidas funcionalidades,

como localización espacial, conectividad o elementos de UI que en otras arquitecturas tendría que implementar desde cero. SonoPhone es versátil ya que no tiene una forma cerrada: en futuras actualizaciones pueden incorporarse otras funciones y corregir los errores que existan. Este Proyecto apenas araña la superficie de todo lo que sería posible hacer en iOS.

Resistencia: Probablemente el punto más débil de SonoPhone, ya que los dispositivos con iOS no son muy resistentes en cuanto a condiciones climáticas se refiere. Además cuentan con una autonomía limitada y sus baterías no son sustituibles.

Facilidad de manejo: Las guías de desarrollo en iOS (ver sección 2.3.3) dejan claro que uno de los principios que debe guiar el diseño de cualquier app es la *usabilidad*. No tiene que haber manuales de usuario. Este es uno de los aspectos en los que la UI de SonoPhone podría mejorarse, ya que, aunque simple, podría ser más intuitiva y atractiva para el usuario.

Coste bajo: Aunque los dispositivos iOS se encuentran en la gama alta de precios en sus respectivos mercados, su enorme popularidad asegura que, si se distribuyera SonoPhone en la App Store, la audiencia potencial de usuarios sería enorme, y éstos no tendrían que pagar por una plataforma específica que solo sirviera para hacer medidas de sonido. En ese sentido, el coste que conlleva la adopción de SonoPhone es muy bajo. Por consiguiente, éste es el gran punto fuerte de un sistema como SonoPhone con respecto a los sistemas de medición de presión sonora ya existentes: *El ciudadano medio no va a comprarse un sonómetro, pero sí lo puede llevar en su móvil.*

En conclusión, este Proyecto ha logrado su objetivo de demostrar que es posible la implementación de un sistema de medida de calidad profesional en el iPhone, si bien la implementación propuesta es embrionaria. El cumplimiento, si no total, al menos en aspectos cruciales de la Norma es técnicamente posible, y proyectos como SonoPhone pueden ser el principio de un cambio fundamental en el campo de la medición acústica.

6.2. Desarrollo futuro

SonoPhone es ante todo un punto de partida de una visión mucho más amplia: la de extender la medición de la presión sonora más allá del ámbito especializado, bajo la premisa de que los dispositivos móviles “inteligentes” lo son porque ayudan a entender

mejor el mundo; ya no son máquinas limitadas a hacer solamente *lo que saben hacer*, sino extensiones del usuario que obtienen datos de su entorno y lo procesan y comunican a otros dispositivos.

Este paradigma se conoce como *Internet Of Things* (IoT), y ha despertado gran interés en los últimos años[1]. La idea que subyace en la IoT es la apertura de nuevos flujos de información que enriquezcan a la sociedad, permitiendo mayor optimización de los procesos productivos y de consumo.

En el caso de SonoPhone, contar con vastas cantidades de datos sobre ruido, contextualizados con su localización y pudiendo observar su evolución temporal, podría permitir una nueva visión sobre el problema de la contaminación acústica.

Actualmente la forma de evaluar el ruido son los mapas de ruido como los que establece la Directiva Europea 2002/49/EC. Estos mapas son "fotos fijas" del nivel de ruido existente en una zona durante un periodo, clasificado en niveles de día, tarde y noche. Estos mapas los elaboran profesionales de la Acústica, equipados con instrumentos de precisión, por orden de los gobiernos, con el objetivo último de hacer cumplir unos niveles establecidos.

En contraste, una base de datos de mediciones de ruido tomadas por los ciudadanos, de forma no organizada y horizontal, permitiría una pluralidad de visiones y análisis, de acuerdo con la filosofía del *Open Data*, en un campo tan complejo como el de la contaminación acústica, en el que confluyen múltiples intereses económicos y personales enfrentados.

En ese sentido, las posibilidades de desarrollo futuro se pueden dividir en mejoras inmediatas de la aplicación y por otro lado evolución de la idea de SonoPhone para convertirlo ya sea en un *sonómetro certificado para iOS*, posible sustituto de los sonómetros actuales, o un sistema recolector de datos sobre ruido de forma semiautomática.

En el primer sentido, algunas de las mejoras que se podrían hacer en la propia aplicación, de cara al lanzamiento de la versión 1.0, son:

- Mejor ajuste a las especificaciones de la norma: Es necesario efectuar varias iteraciones del ciclo evaluación-ajuste-evaluación, con el objetivo de ajustarse lo más posible a las especificaciones de la norma IEC 61672-1 (ver "Precisión" en la sección 6.1). En ese sentido, un Proyecto interesante podría ser una evaluación completa de modelo, que ampliara la evaluación realizada (ver capítulo 5) a otros parámetros especificados por la Norma, como respuesta direccional, respuesta a tren de ondas, otras

[1] Es recomendable la lectura del informe de la ITU al respecto (International Communications Union, 2005), aunque teniendo en cuenta que se publicó *dos años antes de la aparición del iPhone.*

ponderaciones, interferencias electromagnéticas, etcétera.

- UI mejorada: Lo más importante de una app, aparte del buen rendimiento, es la interacción con el usuario. Es necesario dar al usuario más control sobre los parámetros de la medida, incluso en tiempo real, y proporcionar mejor *feedback* visual de lo que se está midiendo. El objetivo es que la aplicación proporcione información de forma clara e inteligible sobre el sonido ambiente. Además se deberían proporcionar elementos visuales personalizados, como logos, imágenes o tipografía, que diferencien la app y la hagan más atractiva visualmente.

- Más medidas: Por ejemplo, una medida muy informativa sobre el ruido ambiente, de uso común en la evaluación del ruido[2], es el contenido en baja frecuencia del sonido. Otro ejemplo son las *componentes tonales* que contenga. Estas medidas implican la implementación de algún tipo de análisis espectral.

- Integración con APIs (Foursquare, Twitter): La naturaleza geolocalizada de las medidas de SonoPhone hace especialmente atractiva su integración con redes sociales centradas en la localización, como Foursquare; los usuarios podrían querer agregar a un sitio información sobre el ruido que hay, o consultarlo antes de acudir. Con Twitter, los usuarios podrían anunciar que han realizado una medición.

- Interacción con medidas realizadas: En una futura evolución de la aplicación, ésta permitiría no solo realizar medidas sino visualizar datos de medidas anteriores, ya sean almacenados en el teléfono o en la nube. Una vista nueva permitiría también comparar la medida que se está realizando con otras ya realizadas.

- Certificación de la app para su uso profesional: Esto implicaría tratar de cumplir todas las especificaciones de la norma IEC 61672, diseñando una batería de *ensayos de evaluación de modelo* y logrando la certificación de un laboratorio de acústica oficial.

En cuanto a posibles proyectos futuros que tuvieran a SonoPhone como punto de partida, se podrían plantear distintos desarrollos, más o menos ambiciosos. Por ejemplo:

- Versión para teléfonos Android / Windows Phone: La extensión a estas plataformas ampliaría extraordinariamente el número de usuarios que podrían utilizar SonoPhone,

[2]Por ejemplo, en las curvas RC (*Room Criteria*).

lo que redundaría en mayor cantidad de datos disponibles, y por tanto aumentaría la utilidad de la base de datos. Este paso no sería muy difícil ya que SonoPhone no exige audio de baja latencia en el dispositivo, requerimiento que actualmente supone una barrera para la expansión de muchas apps de audio profesional fuera de la plataforma iOS.

- Versión tablet (iPad, Android, Windows 8): Un dispositivo con una pantalla táctil varias veces más grande que la del iPhone permitiría mostrar mucha más información sobre las medidas, así como una interacción mucho más amplia con éstas, al tiempo que mantendría las ventajas de movilidad y procesamiento de un *smartphone*.

- Aplicación web para visualizar medidas (gráficas y mapas): Aunque este Proyecto se ha centrado en la parte de cliente, un sistema de medición móvil descentralizado no es posible sin una aplicación de servidor que permita interactuar con los datos almacenados remotamente. Una manera de permitir esto sería el desarrollo de una API web que permitiera extraer datos. Además, se podría desarrollar una UI web, aprovechando tecnologías como HTML5 y Javascript, que permitiera una interactividad semejante a la de las aplicaciones móviles. Otra posibilidad en el marco de las tecnologías de servidor sería, una vez disponible una gran base de datos, el uso de técnicas de minería de datos para extraer patrones en el ruido.

Capítulo 7

Presupuesto del Proyecto

Item	Cantidad	Unidades
Tareas desarrollador		
Documentación previa	100	h
Formación en Desarrollo en iOS	30	h
Desarrollo de la aplicación	100	h
Testeo en simulador	25	h
Corrección de errores	15	h
Testeo en dispositivo iOS	10	h
Mediciones en laboratorio	10	h
Total Horas	290	h
Tarifa desarrollador	25	€/h
Subtotal	7250	€
Costes adicionales		
Uso de iPhone para testeo	10	h
Tarifa uso de iPhone	40	€/h
Subtotal	400	€
Uso del Laboratorio de Electroacústica	10	h
Tarifa laboratorio	100	€/h
Subtotal	1000	€
Costes material (IVA incluido)		
MacBook Pro 13”	1229	€
Vida útil estimada	60	meses
Uso para el Proyecto	6	meses
Coste de uso	122,9	€
Subtotal	122,9	€
Total	8772,9	€
IVA	21 %	
Total (IVA Incluido)	10589,4	€

Tabla 7.1: Presupuesto del Proyecto

Apéndice A

Introducción a la incertidumbre de medida

A.1. Introducción

Este anexo pretende, siguiendo la *Guía para la Expresión de la Incertidumbre de Medida* (Centro Español de Metrología (CEM), 2000), dar una introducción al concepto metrológico de *incertidumbre de medida*, tal como lo aplica la norma IEC 61672, analizada en el capítulo 3, a la hora de especificar las tolerancias de funcionamiento. Puesto que un sonómetro es un instrumento de medida, se aplican conceptos y métodos de la metrología. Entre ellos, hay que contar con la incertidumbre de medida, para que los datos que proporcione tengan sentido. A continuación se expondrán unos conceptos básicos de metrología.

A.2. Conceptos básicos

Una **medición** es un conjunto de operaciones que tienen por finalidad determinar el valor de una magnitud, denominada *mensurando*.

Al efectuar una medición, se considera que se comete un error con respecto al *valor verdadero* del mensurando. Este valor verdadero no puede por su propia naturaleza determinarse con exactitud. La *Guía* considera que los términos "valor verdadero de un mensurando" y "valor de un mensurando" son equivalentes.

El error cometido, que se define como la diferencia entre el valor obtenido y el valor del mensurando, consta de dos errores:

Error sistemático es el error inherente al procedimiento de medida. Se define como la diferencia entre la media que resultaría de un número infinito de mediciones menos el valor del mensurando.

Error aleatorio es el error que se produce en cada medición. Se define como la diferencia entre el valor de la medición y la media que resultaría de un número infinito de mediciones. Su valor esperado es igual a cero.

La *Guía* incide mucho en distinguir los términos "error" e "incertidumbre". El error es un concepto ideal, que no puede ser conocido con exactitud. Mientras que la incertidumbre es un parámetro de la medición, que puede estimarse a partir de datos reales obtenidos. Por ejemplo, el resultado de una medición con elevada incertidumbre puede estar muy cerca del valor verdadero y por tanto tener un error muy pequeño.

Al obtener una medición, se pueden identificar uno o varios *efectos sistemáticos*, que si se consideran significativos con respecto a la exactitud requerida, pueden compensarse aplicando una *corrección*. Sin embargo, dicha corrección, al no poder conocerse con exactitud el valor del error sistemático cometido, aporta a su vez incertidumbre a la medida. Una vez aplicada dicha corrección, se obtiene una *estimación*

A.3. La incertidumbre de medida

La *incertidumbre de medida* es una estimación del campo de valores dentro del que se halla el valor verdadero del mensurando. La incertidumbre expresa la duda de que la medición obtenida represente el verdadero valor del mensurando. Las fuentes de incertidumbre incluyen: muestreo imperfecto del mensurando, efecto imprevisto de las condiciones ambientales, resolución finita de los instrumentos, o efectos sistemáticos no tenidos en cuenta, así como el efecto de la corrección de éstos.

La *Guía* da una definición operativa: la incertidumbre de medida es un parámetro que caracteriza la dispersión de los valores que pudieran ser atribuidos al mensurando. Se considera que el resultado de la medición es la mejor estimación del valor del mensurando, y que todos los componentes de la incertidumbre de medida añaden dispersión.

Puesto que se trata con variables aleatorias, es necesario el uso de herramientas estadísticas para determinar y evaluar la incertidumbre de medida. En ese sentido, se definen:

Incertidumbre típica: es la incertidumbre de una medición, expresada en forma de des-

viación típica.

Incertidumbre típica combinada: es la obtenida como la raíz cuadrada de la suma ponderada de las varianzas o covarianzas de otras magnitudes.

Incertidumbre expandida: es una magnitud que define un intervalo alrededor del resultado de una medición, en el que se espera encontrar una fracción importante de la distribución de valores atribuibles al mensurando. Está relacionada con el concepto estadístico de *nivel de confianza* asociado al intervalo.

Factor de cobertura: es un factor por el que se ha de multiplicar la incertidumbre típica combinada para obtener la incertidumbre expandida.

Para obtener las varianzas a partir de las que se calcula la incertidumbre típica combinada, es necesario asumir hipótesis sobre la distribución estadística de los resultados de la medida. Existen dos tipos de evaluación de dichas varianzas:

- La evaluación tipo A se basa en la estimación de la varianza a partir de observaciones repetidas, a partir de la distribución de frecuencia observada.
- La evaluación tipo B se basa en una función de densidad de probabilidad supuesta (probabilidad subjetiva).

Para controlar y cuantificar la incertidumbre, es importante disponer de un modelo matemático del mensurando como función de otras magnitudes. Haciendo variar estas magnitudes y observando la variabilidad de la medición, se puede estimar la sensibilidad de la medición a la variación de las magnitudes de entrada, así como si éstas son independientes o están correlacionadas.

El procedimiento estadístico detallado para obtener la incertidumbre se aclara en la *Guía*, y está fuera del alcance de este Anexo.

A.4. Aplicación a la norma IEC 61672

En la norma IEC 61672, todas las tolerancias, por ejemplo para la respuesta direccional, a tren de ondas o ponderaciones frecuenciales (ver sección 3.4) se indican incluyendo la incertidumbre expandida de la medida con un factor de cobertura de 2, correspondiente

aproximadamente a un nivel de confianza del 95 %[1]. Así, el fabricante del sonómetro debe restar de la tolerancia dada el valor de la incertidumbre expandida proporcionada en el anexo A para obtener el objetivo de diseño y fabricación. El laboratorio que realice el ensayo de evaluación de modelo debe, al realizar sus mediciones, controlar que la incertidumbre no exceda lo especificado en la norma.

[1]Este valor se obtiene suponiendo que la incertidumbre sigue una distribución normal, en cuyo caso el 95 % de los valores está comprendido en dos desviaciones típicas.

Apéndice B

Conceptos de desarrollo en iOS

A continuación se introduce lo necesario para entender el código de la aplicación. Se presupone conocimiento de conceptos de programación orientada a objetos.

B.1. El lenguaje Objective-C

Objective-C es un lenguaje de programación orientado a objetos creado a principios de los años 80. Se trata de un *superconjunto estricto* de C, es decir que todas las instrucciones de C son válidas en un programa Objective-C. Muchas de las características propias del lenguaje están inspiradas en Smalltalk, uno de los primeros lenguajes orientados a objetos. Apple lo empezó a utilizar después de que la compañía NeXTStep, fundada por Steve Jobs, desarrollara el lenguaje y creara un compilador para sus estaciones de trabajo. Después, Apple adquiriría NeXTStep e incorporaría su tecnología. Esta evolución histórica se conserva en el uso del prefijo `NS` en muchas clases de la API fundamental, Foundation.

El código Objective-C se escribe en archivos `.m` y `.h` para la la *implementación* e *interfaz* de la clase, respectivamente. La interfaz declara los métodos y propiedades públicos de la clase, indicado por la palabra clave `@interface`, y la implementación contiene el código, marcado por `@implementation`. También puede definirse una interfaz privada para la clase.

Los métodos pueden ser de instancia (indicados por el signo −) y de clase (indicados por +). La sintaxis de los métodos intercala el nombre del método (llamado *selector*) con los parámetros, por ejemplo

```
- (id)initWithContentsOfFile:(NSString *)path
 encoding:(NSStringEncoding)enc error:(NSError **)error
```

es la declaración de un método de instancia (de la clase `NSString`) que toma tres parámetros. El selector sería `initWithContentsOfFile:encoding:error:`.

Se ofrecen características similares a otros lenguajes orientados a objetos, como herencia, protocolos, y propiedades con sus correspondientes accesores[1]. Algunas de las características peculiares de Objective-C en comparación a otros lenguajes orientados a objetos como C++ o Java son:

Mensajes En Objective-C, no se llama a un método de una instancia, sino que se pasan mensajes con una sintaxis `[receptor mensaje]`. En tiempo de ejecución, se busca en el receptor un método con el mismo nombre del mensaje y se resuelve la llamada. Si el receptor no responde a ese mensaje, da una excepción. Si el receptor es un puntero no asignado (`nil`), no hace nada.

Tipos dinámicos Aunque se pueden utilizar los tipos y estructuras de C, los objetos, tanto de las APIs como definidos por el desarrollador, heredan de `NSObject`. Los objetos se manejan como referencias (punteros) y se definen por los mensajes que pueden recibir. Una palabra clave especial, `id`, representa un puntero genérico. Esto permite mayor flexibilidad y bajo acoplamiento entre clases. Junto con los protocolos, los punteros opacos forman la base del patrón de delegación, descrito en la sección B.4.

Introspección Objective-C hace un uso muy abundante del *runtime*, aplazando muchas decisiones hasta la ejecución del programa en vez de tomarlas al desarrollo. Para lidiar con la flexibilidad de los tipos dinámicos y el sistema de mensajes, se permite "preguntar" a una instancia si puede responder a un determinado selector, pasando como parámetro el nombre de éste. En su forma sofisticada, esto permite la llamada codificación clave-valor (KVC), es decir, acceder a las propiedades de un objeto por su nombre en vez de mediante un accesor. También puede evaluarse si una clase implementa un determinado protocolo o es subclase de otra determinada.

Manejo de memoria A diferencia de Java o .NET, Objective-C no tiene recolección de basura. El programador tiene la responsabilidad de asignar memoria (`alloc`) para los objetos que utilice y de liberarla (`release`) cuando ya no sea necesaria. Sin embargo, a partir de XCode 4.2 e iOS 4, se introdujo ARC, un sistema de manejo de memoria. ARC significa Conteo Automático de Referencias, y elimina la necesidad

[1] Muchas diferencias con otros lenguajes OO son de nomenclatura. Por ejemplo se habla de protocolos en vez de interfaces, o de inicializadores en vez de constructores.

de realizar las liberaciones de memoria manualmente. Se basa en el principio de mantener un control de cuántas referencias tiene un objeto; cuando el contador de referencias llega a cero, se libera la memoria. Hay dos tipos de referencia, débil (`weak`) y fuerte (`strong`). Las referencias débiles no aumentan el contador de referencia, y se introducen para evitar que dos objetos se referencien mutuamente de forma fuerte y por tanto su contador nunca llegue a cero, condición conocida como referencia circular.

Hay que matizar que este manejo automático de memoria se aplica a los objetos de Objective-C. En iOS se puede también utilizar cualquier construcción de C estándar, incluyendo funciones que asignan y liberan memoria como `malloc` y `free`. En este caso el manejo de memoria sigue siendo responsabilidad del desarrollador. El framework Core Foundation, escrito en C, proporciona "puentes" de sus tipos a los objetos de Objective-C, que permiten que ARC maneje su memoria. El funcionamiento de este sistema, sin embargo, está fuera de los objetivos de esta Memoria.

B.2. El entorno de desarrollo

Xcode es el entorno de desarrollo integrado (IDE) proporcionado por Apple para el desarrollo de aplicaciones de iOS y OS X. Integra un editor avanzado de código fuente con autocompletado y revisión de sintaxis, diseñador de interfaces gráficas (*Interface Builder*), compilador y enlazador, capacidades avanzadas de depuración y análisis de rendimiento, entorno de pruebas unitarias, y un simulador de dispositivos iOS, entre otros. Al crear un nuevo proyecto iOS, se pueden elegir varias plantillas. Todas ellas generan aplicaciones perfectamente funcionales, aunque vacías. Para empezar a desarrollar basta construir la interfaz e implementar la lógica del controlador (ver sección B.5). En la figura B.1 se puede ver la pantalla de entrada que presenta XCode cuando se crea un nuevo proyecto.

El *Organizer* de XCode agrupa la documentación (de iOS, OS X y el propio XCode), un organizador de proyectos, los repositorios que se usen, los dispositivos iOS en los que se puede probar la aplicación e incluso mandar la app para su revisión a la App Store. XCode es totalmente gratuito, sin más requerimiento que el registro (también gratuito) como desarrollador en la comunidad Apple Developer.

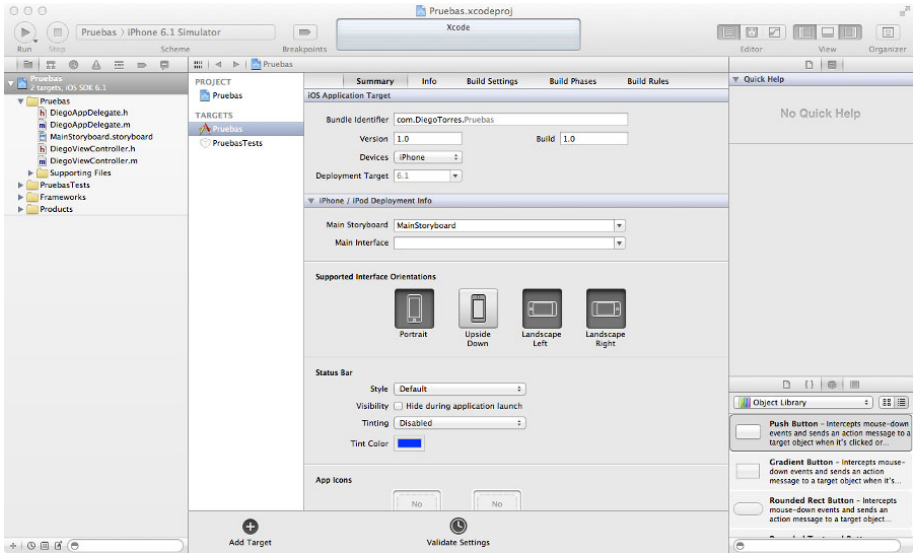

Figura B.1: Pantalla de entrada al crear un proyecto en XCode

B.3. El patrón MVC

El patrón MVC, siglas de **M**odelo **V**ista **C**ontrolador, es uno de los patrones de diseño más importantes en el desarrollo de software, y en Cocoa para iOS u OS X su uso es prácticamente obligatorio. Se basa en la división de responsabilidad entre las distintas partes de una aplicación, y no se refiere a ninguna práctica de programación en concreto sino a la planificación global de la aplicación. Así, existen maneras muy distintas de llevar a la práctica el patrón. Aquí se aplicará la manera que usa Cocoa.

El Modelo Representa la parte de la aplicación que se ocupa de los datos. Un ejemplo habitual es una clase que interactúa con una base de datos, o que descarga y sube contenidos a la red. El modelo debe ser totalmente independiente de cómo se presentan los datos que maneja.

La Vista Representa la parte de la aplicación que maneja la interacción con el usuario. Recibe datos y los presenta al usuario. También recoge las acciones del usuario y las reenvía a la parte del código adecuada. Ejemplos de vistas son la ventana de una aplicación en OS X, o el HTML generado por una aplicación web. Una vista debe ser estar compuesta de elementos lo más *reutilizables* posible, para asegurar una apariencia coherente en todo el SO.

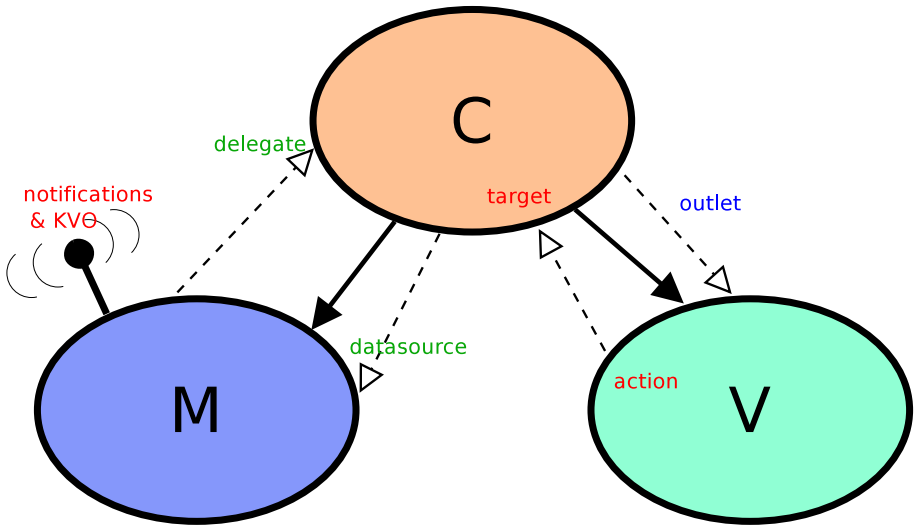

Figura B.2: Esquema de la implementación Cocoa del patrón MVC

El Controlador Cumple el rol de mediador entre la Vista y el Modelo, pudiendo interactuar con otros controladores. Pide datos al Modelo cuando es necesario, los procesa y los envía a la Vista para que se muestren al usuario, siendo también capaz de responder a las acciones del usuario sobre la UI.

En Cocoa, el patrón se realiza en forma de relaciones entre clases, como se muestra en la figura B.2. En el esquema se muestra la relación MVC más simple posible, ilustrando alguna de las relaciones que se pueden dar. En primer lugar, el controlador "posee" las instancias del modelo y la vista, es decir que tiene una referencia *fuerte* a ellos (flechas sólidas). La vista estaría formada por elementos de la UI[2], y el Controlador sería un controlador de vista (`ViewController`).

El controlador muestra su información a la vista a través de sus salidas (*outlets*). Estos se conectan gráficamente en el Interface Builder; se puede entonces acceder al objeto de la UI en cuestión como una propiedad del Controlador, marcada como `IBOutlet`.

Para controlar las acciones del usuario sobre la UI, se usa el patrón diana-acción (*target-action*). Al dispararse un evento en la capa de Vista, se "hace diana" sobre un método del Controlador que se ha designado como `IBAction`.

[2]En el caso más general, una vista puede ser personalizada sobreescribiendo el método que la renderiza, drawRect.

En cuanto a la comunicación entre Modelo y Controlador, se suele conseguir utilizando el patrón delegación, como se explica en la sección B.4. De esta manera el Modelo no está atado a una clase de Controlador en particular. El Controlador se establece como delegado del Modelo y puede recibir información de éste. Una limitación de este enfoque es que solo una instancia puede ocupar el puesto de delegado a la vez. Para solucionar eso, Cocoa provee un mecanismo de notificaciones (representado como una antena en la figura) que permite a una clase anunciar a todos aquellos que "sintonicen" (declarando que se suscriben al notificador) la señal más oportuna. Otro mecanismo, la observación clave-valor (KVO), permite suscribirse a los cambios solo en una propiedad de un objeto.

Un punto fundamental del diagrama es que el Modelo y la Vista no tienen ninguna clase de relación directa entre sí. El objetivo es maximizar la *reusabilidad* de estos componentes, encapsulando la lógica propia de la aplicación en los Controladores.

B.4. El patrón de delegación

Otro de los patrones más ampliamente utilizados por los frameworks del SDK y que subyace tras la implementación del patrón MVC, es el patrón delegación. En este patrón, un objeto ofrece a otro la posibilidad de convertirse en su delegado. La clase que delega puede mandar mensajes al delegado para que actúe en consecuencia, pero desconoce la implementación de éste.

Para entender este patrón es fundamental el concepto de *protocolo*. Un protocolo es un "contrato de operaciones" que una clase declara cumplir. En Objective-C, pueden contener declaraciones de métodos que han de implementarse obligatoriamente y otros que son facultativos.

La relación entre el objeto que delega y el delegado es de *bajo acoplamiento*, puesto que se realiza mediante un protocolo; la clase solo necesita saber que el delegado es capaz de responder a determinados mensajes, los que constan en el protocolo. Esto promueve la reusabilidad del código.

Una fuente de datos (*datasource*) es una aplicación del patrón delegación en el que lo que se delega son los datos; el objeto llama al método del delegado que devuelve los datos necesarios. Por ejemplo, el protocolo `UITableViewDataSource` contiene un método obligatorio que devuelve una celda en la posición especificada.

B.5. Anatomía de una app

La aplicación iOS más simple posible consta por lo menos de una vista, su controlador correspondiente y algún tipo de modelo. La infraestructura, ilustrada en la figura B.3 comprende un objeto `UIApplication` que crea el *delegado de aplicación*. Éste se encarga de manejar los eventos que se produzcan y de recibir notificaciones del sistema respecto a los cambios de estado de la aplicación, que puede estar activa, inactiva, en segundo plano...

En iOS solo existe una "ventana" a la vez, representada por un objeto `UIWindow`, que controla las vistas. Un `ViewController` puede contener a otros `ViewController` y controla una jerarquía de vistas, que pueden ser controles estándar de UIKit o vistas personalizadas dibujadas por el código del desarrollador.

La UI, con sus controladores y vistas, está organizada en un *storyboard*. En esta metáfora cinematográfica, el conjunto de vistas que ocupa la ventana en un momento dado se llama *escena*, y entre ellas se definen transiciones (*segues*). iOS proporciona varios "controladores de controladores", como `UINavigationController`, que navegan entre varios subcontroladores, cada uno con sus vistas.

B.6. Core Audio

Core Audio es el conjunto de frameworks que se encargan del audio en iOS y OS X. Mientras que otros frameworks, como AV Foundation, gestionan el audio y vídeo como recursos (*assets*), Core Audio (en adelante CA) permite trabajar a nivel de las muestras de audio. Muchas de las APIs de CA están en C, aunque gracias a que Objective-C es un superconjunto de C (ver sección B.1), la integración es inmediata.

CA está formado por diferentes capas que manejan distintos niveles de abstracción. Al nivel más bajo está la HAL (*Hardware Abstraction Layer*), con la que no se interactúa directamente, sino que se "extrae" de ella el audio (para entrada) o se le proporciona audio sintetizado o leído para la salida. Dos APIs permiten realizar esa operación de extracción e inyección de audio:

Audio Queue Services (AQ) Se basa en una operación tipo "cinta transportadora". Para implementar una AQ de *entrada*, se inicializa la cola y se le pasan uno o varios *buffers* vacíos. AQ va rellenando los buffers y llama a una función definida por el desarrollador (*callback*) para que pueda "vaciar" el buffer, procesar los datos de audio y

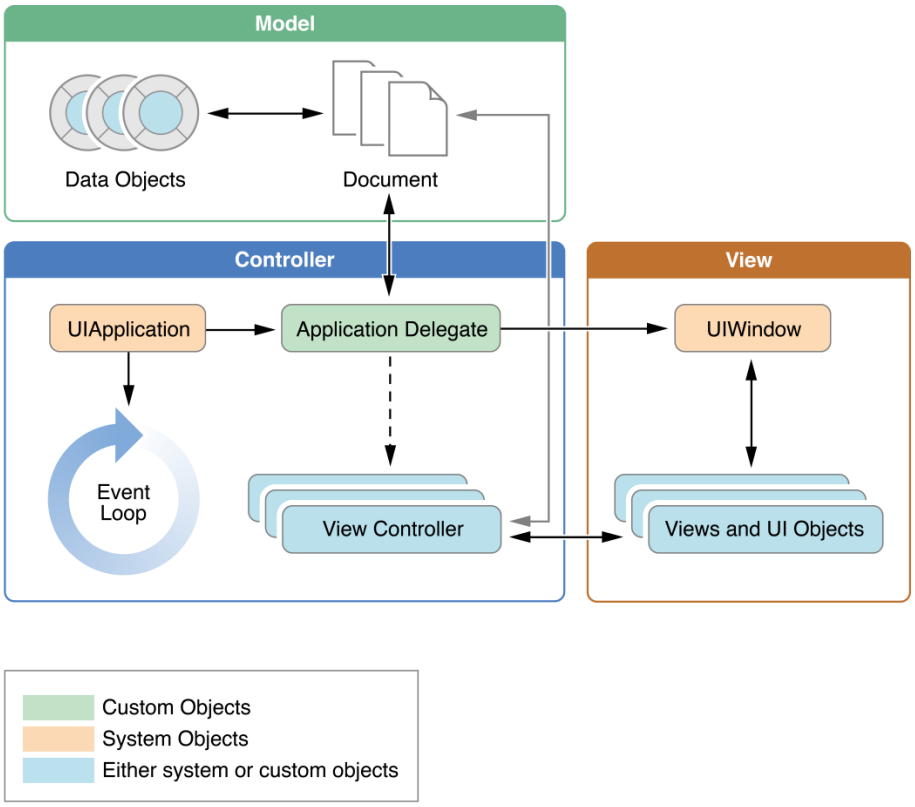

Figura B.3: Estructura básica de una app. Extraído de Apple Inc. (2013a), ©Apple Inc.

volver a poner el buffer en la cinta (`EnQueue`). Una AQ de salida funciona de forma complementaria: en el callback, es el código de usuario el que rellena el buffer con datos para que la AQ los transporte a la salida de audio.

Audio Unit Services (AU) Es una API de más bajo nivel pensada para el procesado en tiempo real. Son bloques de procesado con diversas funciones: de entrada, de salida, de efectos, de mezcla... que se conectan en una estructura dinámica conocida como *grafo*. La programación con Audio Units es compleja y su uso solo está recomendado en aplicaciones que requieran el máximo rendimiento de audio.

En Core Audio para iOS, a diferencia de en OS X, las muestras de audio no se manejan en formato de coma flotante, lo cual permite el procesado con menos gasto de batería. Se usan dos formatos: PCM lineal con muestras enteras de 16 bits para AQ, y un formato LPCM de coma fija de 32 bits (8.24) para AU. Estos formatos se conocen como *canónicos*.

B.6.1. Audio Queue Services

En SonoPhone se ha optado por basar el procesado de audio en AQ, ya que la aplicación no tiene salida de audio ni requerimientos de procesado de baja latencia que justificarían la complicación extra de usar AU.

El procedimiento de obtención y procesado del audio con AQ es como sigue:

- Se declara una estructura C que contendrá el *estado* de la aplicación, es decir, todo lo que se desee tener disponible en el código del callback. Esto es necesario puesto que AQ Services no sabe nada de la clase que lo invoca, por tanto no se puede acceder a las propiedades ni en general el estado interno de la clase. En la estructura se puede incluir el formato de audio (ver siguiente ítem), indicadores binarios (*flags*) que indiquen por ejemplo si la AQ está activa, etcétera. La estructura aparece como uno de los parámetros del callback, en forma de puntero sin tipo, `void *`.

- Se define el formato de audio deseado, rellenando una estructura de tipo `AudioStreamBasicDescription`, que contiene todos los campos necesarios: frecuencia de muestreo, bytes por cuadro[3], número de canales, etcétera. Para ello, se interroga a Audio Session (ver sección B.6.2) por las capacidades del dispositivo. También hay que calcular el tamaño correcto de los buffers para que puedan contener la suficiente longitud de audio.

- Se escribe el callback, una función C cuya declaración debe coincidir con la de `AudioQueueInputCallback`.

- Se crea la AQ, mediante la función `AudioQueueNewInput`, a la que se le pasa como parámetros la estructura que contiene el formato, un puntero al callback, un puntero a la estructura de estado y dos parámetros que indican cómo se ejecutará el callback — el valor por defecto es que se ejecuta en un hilo separado del principal, para no bloquear la UI.

- Se asigna memoria a los buffers y se les pone en cola, preparados para recibir muestras de audio.

- Se pone en marcha la AQ, que empieza a llamar al callback en cuanto va llenando cada buffer.

[3]Se define un cuadro (*frame*) como la unidad mínima de audio; consta de tantas muestras como canales. En formatos comprimidos, se define también el paquete (*packet*), que puede contener varios cuadros.

B.6.2. Audio Session

Audio Session es una API, parte de Core Audio, que gestiona la asignación de audio a las aplicaciones. Cuando una app quiere reproducir o grabar audio, solicita al sistema la apertura de una sesión. iOS usa la analogía de la torre de control: la aplicación solicita el uso del audio el sistema va activando o desactivando las sesiones según sea necesario. Audio Session también provee a la aplicación de un método para reaccionar a las interrupciones del audio, por otras aplicaciones, bloqueo del teléfono o llamada entrante, por ejemplo.

Existen varias categorías de sesión de audio que una app puede solicitar, según necesite reproducción continua aun cuando el teléfono esté bloqueado, grabación, u otros. En el caso de SonoPhone, se utiliza la categoría `kAudioSessionCategory_RecordAudio`, que asegura que no se para cualquier reproducción de audio mientras la app esté utilizando la entrada de sonido.

Además de las categorías, existen varios *modos* de funcionamiento, que indican al sistema si debe efectuar procesado de audio de algún tipo en la entrada o no, por ejemplo ajuste de ganancia, cancelación de ruido o ecualización para mejorar la calidad de la voz. Para este Proyecto se necesita el audio más puro posible, así que se ha usado el modo *Measurement*, que trae el sonido "tal cual".

Apéndice C

Pruebas de funcionamiento

Para realizar la prueba de conformidad con la norma citada en la sección 5.1, se obtuvo experimentalmente la respuesta al impulso de la entrada de audio mediante el método de secuencias de longitud máxima (*Maximum Length Sequence*, MLS).

C.1. Justificación teórica

Las secuencias de longitud máxima son un tipo de señal binaria y periódica, que sólo contiene los valores 1 y -1, de longitud $P = 2^m - 1$, donde P es el periodo y m es un entero que representa el orden de la secuencia. Tienen una serie de propiedades que la hacen atractiva para el cálculo de la respuesta al impulso de un sistema (Thomas, 2009):

- Autocorrelación igual a un impulso: la función de autocorrelación de una MLS es igual a 1 para el retardo cero y $1/P$ para todos los demás.
- Respuesta en frecuencia plana: las MLS comparten, a pesar de ser señales completamente deterministas, varias propiedades estadísticas con el ruido blanco, por ejemplo la respuesta en frecuencia plana.
- Fácil generación y análisis: Las MLS se pueden generar de forma muy eficiente con registros de desplazamiento, y su análisis también es simple en términos computacionales.

La periodicidad de la MLS implica que lo que se obtiene al estimular el sistema es la *respuesta al impulso periódica*, es decir la convolución circular de la respuesta del sistema

con la MLS. Por tanto, hay que elegir P mayor que la longitud de la respuesta al impulso del sistema, para prevenir la aparición de *aliasing*.

Se debe estimular el sistema bajo estudio con varias MLS idénticas consecutivas, para despreciar efectos debidos a que el sistema no estuviera excitado inicialmente. Además, si se toman varias medidas en consideración al calcular la respuesta, se ganan 3 dB de relación señal-ruido cada vez que se dobla el número de medidas.

Para calcular la respuesta al impulso a partir de la salida del sistema y la MLS con la que se estimuló, se ha de realizar la siguiente operación:

$$h'(n) = \frac{1}{P-1}\left(\sum_{k=1}^{P} y(k)x(k-n)\right) \tag{C.1}$$

donde $h'(n)$ es la respuesta periódica al impulso, $y(k)$ la señal de salida obtenida y $x(n)$ la MLS.

La implementación de esta operación de forma computacionalmente eficiente comporta el uso de técnicas sofisticadas como transformadas de Hadamard, que están fuera del alcance de este Proyecto. Se usó en la práctica la función `tfestimate` de MATLAB, que, usando el método de Welch, estima directamente la respuesta en frecuencia a partir de la entrada y salida (ver sección C.3).

C.2. Montaje experimental

Para medir la respuesta de la entrada de audio del iPhone, se estimuló el sistema con una señal eléctrica, eliminando de esta forma los efectos acústicos del micrófono, altavoz, sala, etcétera.

Se preparó el iPhone para grabar con una pequeña aplicación desarrollada para la ocasión, llamada *SonoRecorder*, que incorpora una versión reducida de la clase `SonoModel` (ver sección 4.2), que, en vez de escribir los datos de las medidas, graba el contenido de los buffers en un archivo de audio en formato `.wav`.

Por otra parte, se generó en MATLAB[1] una señal MLS de 15 bits y 20 repeticiones, que se almacenó en un archivo `.wav` en el ordenador.

[1] Usando el excelente "Dual channel MLS Analyzer" desarrollado por el Departamento de Acústica de la Universidad de Oldenburg, y cuyo código MATLAB se encuentra en `http://www.akustik.uni-oldenburg.de/34940.html` (última consulta: Julio 2013).

Se conectó mediante un cable de audio la salida de audio del ordenador a la entrada del iPhone, utilizando un adaptador iRig para adaptar el cable *jack* de 1/4" a la entrada TRRS del iPhone.

El procedimiento de medida consistió en:

- Puesta en marcha de la grabación en el teléfono.
- Puesta en marcha de la reproducción de la señal MLS en el ordenador.
- Parada de la grabación en el teléfono.
- Recolección del archivo `.wav` grabado por la aplicación, mediante iTunes (ver sección 4.7).

C.3. Procesado

Una vez obtenida la grabación de la respuesta a la MLS, se procedió a extraer de ésta la parte útil con un programa de edición de audio[2], y se cargó en MATLAB junto con la MLS original. El siguiente paso fue igualar las longitudes de ambas señales a la menor de las dos; puesto que el orden de la MLS es 15, la longitud de la señal es $2^{15} - 1$, es decir de $P = 32767$ muestras. La señal originalmente tenía 20 repeticiones, y de la señal de salida recortada se obtuvieron 19 repeticiones útiles.

Se calculó la autocorrelación de la MLS de entrada y la correlación cruzada de ésta con la salida, para cada segmento de longitud P, y posteriormente se promediaron todos los cálculos en dos vectores. Estos vectores se pasaron como argumentos a la función `tfestimate` del *Signal Processing Toolbox* de MATLAB, especificando el uso de una ventana Blackman de 512 muestras de longitud[3]. El procedimiento de obtención de las correlaciones y la estimación de la respuesta en frecuencia se ilustran en los listados C.1 y C.2, respectivamente. La Norma IEC 61672 exige que se proporcionen datos de la desviación de la respuesta del sistema con respecto a la respuesta de referencia con ponderación A, de forma tabular, para las bandas de tercio de octava de 63 Hz a 8 kHz. La respuesta obtenida, en función de la frecuencia, se muestra en la figura C.1. Sin embargo, para obtener datos que se puedan comparar con la especificación, se debe promediar en las bandas de tercio

[2] Audacity, editor de audio avanzado *open source*.

[3] Esta función estima la función de transferencia por el método de Welch, es decir mediante el uso de periodogramas de segmentos enventanados de la señal.

Listado C.1: *Script* de MATLAB para procesar los resultados de la medición

```
fs = 48000;
mls_orig = wavread('MLS_original.wav');
mls_2 = wavread('MLS_2_recortada.wav');

len_mls_orig = length(mls_orig);
len_mls_2 = length(mls_2);

od = 15; % Orden de la MLS
len = 2^od-1; % longitud de la MLS = 32767
im1 = zeros(1,2*len-1);
im2 = zeros(1,2*len-1);
% Tratamos de encontrar la longitud
% mínima entre los 2, múltiplo de len
rep = floor(min(len_mls_orig,len_mls_2)/len); % rep = 19
% rep es el número de repeticiones de la MLS disponible
for k=2:rep
    st = (k-1)*len+1;
    en = k*len;
    % autocorrelación:
    [tmp1,c] = xcorr(mls_orig(st:en),'unbiased');
     % correlación salida/entrada:
    [tmp2,c] = xcorr(mls_2(st:en),mls_orig(st:en),'unbiased');
    im1 = im1+tmp1';
    im2 = im2+tmp2';
end

% Calcula la media entre todas las correlaciones:
im1 = im1./rep;
im2 = im2./rep;
```

Listado C.2: Estimación de la respuesta en frecuencia

```
% im1 = autocorrelación de la entrada (MLS)
% im2 = correlación cruzada salida-entrada
% fs = frecuencia de muestreo
winlen = 512;
wind = blackman(winlen); % Ventana Blackman
[T,f] = tfestimate(im1,im2,wind,0,length(im1),fs);
```

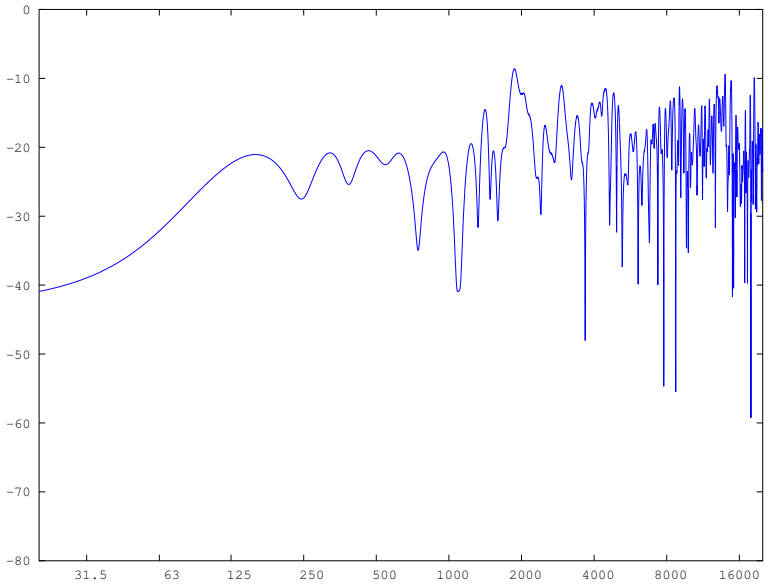

Figura C.1: Respuesta en frecuencia estimada.

de octava especificadas, obteniendo a partir de la frecuencia central de la banda i-ésima f_{Ci} las frecuencias superior e inferior, $f_{Hi} = 2^{1/6} f_{Ci}$ y $f_{Li} = 2^{-1/6} f_{Ci}$, y promediando los valores entre estas frecuencias. El *script* que se muestra en el listado C.3 se aprovecha de la funcionalidad de indexado lógico de MATLAB para extraer la respuesta en bandas de tercio de octava. Una vez obtenidos los valores en bandas, es necesario convertir el resultado a decibelios. Con el fin de poder comparar con los valores que da la norma, se elige como referencia el valor a 1 kHz, que en cualquier ponderación ha de ser igual a 0 dB.

C.4. Determinación de la incertidumbre de medida

La norma IEC 61672, en su parte 2 (UNE-EN, 2005), referente a los *ensayos de evaluación de modelo*, especifica que para calcular la incertidumbre expandida de medida, hay que tener en cuenta por lo menos cinco fuentes de incertidumbre:

- Incertidumbre procedente de la calibración de los instrumentos individuales, incluyendo la incertidumbre asociada a los calibradores.

- Incertidumbre asociada a las condiciones ambientales.

Listado C.3: *Script* de MATLAB para obtener la respuesta en bandas

```
function [Y,bandas]=calculatebandlevel(X,f)
% calcula el nivel en bandas de tercio de octava
% T es un vector de datos en función de la frecuencia
% f es un vector de las frecuencias correspondientes
bandas = [16 20 25 31.5 40 50 63 80 100
    125 160 200 250 315 400 500 630 800
    1000 1250 1600 2000 2500 3150 4000
    5000 6300 8000 10000 12500 16000 20000]';
n = length(bandas);
cl = 2^(-1/6);
ch = 2^(1/6);
Y=zeros(n,1);
for i=1:n
    fc = bandas(i); % f. central
    fl = fc * cl; % f. límite inferior
    fh = fc * ch; % f. límite superior
    Y(i) = mean(X(f > fl & f < fh));
end
```

- Incertidumbre en las señales aplicadas.
- Incertidumbre aleatoria asociada a la repetición de las medidas.
- Incertidumbre ligada a la resolución limitada del dispositivo de presentación de resultados.

Cada uno de estos valores toma la forma de una desviación típica, σ_i. Para determinar la incertidumbre expandida, según lo explicado en el apéndice A, se calcula la suma cuadrática de todas ellas y se multiplica por el factor de cobertura elegido, C.

Una vez determinada la incertidumbre, debe compararse su magnitud con los valores dados en el apéndice A de la norma IEC 61672-1 (ver sección 3.8). Si el valor excede lo especificado, la medida no puede darse por válida, y el laboratorio que realiza la evaluación no puede dar la aprobación en ese campo. Los valores máximos de incertidumbre de medida que especifica la norma se proporcionan en la tabla C.1. Para la medida realizada, la incertidumbre puede considerarse como proveniente de tres componentes:

- Incertidumbre introducida por el hardware (σ_h): La salida del audio del ordenador,

Rango de frecuencias (Hz)	Incertidumbre expandida máxima (dB)
10 - 200	0,5
200 - 1250	0,4
1250 - 10000	0,6
10000 - 20000	1,0

Tabla C.1: Incertidumbres máximas de medida, según la Norma.

el cable utilizado y el adaptador iRig puede considerarse que tienen respuesta plana en el rango de frecuencias de interés. Sin embargo, al hacer esta suposición se comete una imprecisión, que se cuantifica como una incertidumbre de 0.1 dB.

- Incertidumbre debida a la estimación (σ_e): La respuesta obtenida es solamente una estimación, y por tanto tiene una incertidumbre asociada. Puesto que el método de Welch se basa en la FFT, existe un compromiso entre la resolución frecuencial y temporal. En este caso, se ha elegido una longitud de FFT bastante grande, de manera que la incertidumbre frecuencial es pequeña, con un valor estimado de 0.2 dB.
- Incertidumbre asociada a la repetibilidad de la medida (σ_r): Esta incertidumbre proviene del hecho de que, al repetir una medida en las mismas condiciones, los resultados no van a ser nunca exactamente iguales, debido a las variaciones incontrolables en las condiciones ambientales. Sin embargo, al tratarse de una medida puramente eléctrica, la influencia de éstas es pequeña (menor que lo que sería en una medida acústica), y por tanto la incertidumbre asociada es pequeña, del orden de 0.1 dB.

A partir de estas estimaciones, se estima la incertidumbre expandida como ya ha sido explicado, aplicando un factor de cobertura de 2[4]:

$$I = 2\sqrt{\sigma_h^2 + \sigma_e^2 + \sigma_r^2} = 0{,}5 \quad \text{dB} \tag{C.2}$$

Este valor, que es una *cota inferior* de la incertidumbre cometida a cualquier frecuencia, es superior a los valores especificados en la tabla C.1 para el rango de frecuencias de 200 a 1250 Hz. Por tanto, aunque esta medida no sería válida para la evaluación de modelo, sí da una idea de la respuesta al mantener un valor de incertidumbre bastante cercano al dictado por la Norma.

[4]Correspondiente a una confianza del 95 %, según se indica en el apéndice A de la norma.

Apéndice D

Developer Program de la UPM

Como se comentó en la sección 2.3.1, para el desarrollo de aplicaciones en iOS es prácticamente obligatoria la inscripción del desarrollador en uno de los *Developer Program* ofrecidos por Apple. El propósito de esto es doble: por un lado, el desarrollador y sus aplicaciones quedan identificados criptográficamente, de manera que se asegura que la aplicación no se modifica sin el conocimiento del desarrollador. Por otro lado, Apple se asegura de que la única forma de que los desarrolladores distribuyan sus apps es la App Store, que está completamente controlada por la compañía.

D.1. Funcionamiento del Developer Program

D.1.1. Certificados

Cuando un desarrollador se registra como Miembro de un programa, se genera un certificado de seguridad personal, que tiene forma de cifrado asimétrico, con una clave pública y una privada. Con estas claves se firma el código de la aplicación, de manera que queda identificado unívocamente.

La firma del código es imprescindible para enviar la aplicación a la App Store, y para testear ciertas funcionalidades, como iCloud o compras dentro de la aplicación (*In-App purchasing*).

D.1.2. Perfiles de aprovisionamiento

Para probar la aplicación en un dispositivo, es necesario realizar un proceso llamado aprovisionamiento (*provisioning*), en el que se crea un perfil que asocia un *grupo de trabajo* con un dispositivo concreto. En el caso de una suscripción personal al Developer Program, el equipo de trabajo está formado por una única persona, pero en el caso de una suscripción empresarial o universitaria, el administrador da de alta a los desarrolladores y sus dispositivos en grupos. De esta manera, cualquier desarrollador del grupo puede probar cualquier aplicación firmada por algún miembro en cualquiera de los dispositivos aprovisionados en el grupo.

Un perfil de aprovisionamiento (*provisioning profile*) consta de los siguientes elementos únicos:

- Un dispositivo. Los dispositivos se identifican por su UUID, aunque se nombran por su nombre[1].
- Un identificador de lote (*bundle identifier*). Es una clase de identificador de recurso que ha de especificarse obligatoriamente al crear la aplicación. Así, la aplicación queda identificada por el identificador de lote, más el nombre de la aplicación[2]. De esta manera, se permiten probar todas las aplicaciones que tengan el identificador de lote especificado.
- Un desarrollador. Se identifica por su certificado, como se explica en la sección D.1.1.

D.2. El Developer Program de la UPM

Existen tres tipos de Developer Program:

Development Program estándar Pensado para individuos u organizaciones que piensan distribuir aplicaciones en la App Store. Es el tipo de programa más común.

Enterprise Program Pensado para empresas que quieren desarrollar aplicaciones iOS para su distribución y uso exclusivamente dentro de la compañía.

University Program Se ofrece gratuitamente a las universidades que quieran impartir cursos de desarrollo para iOS.

[1] Por ejemplo: "iPhone de John Appleseed"

[2] En el caso de SonoPhone, es `com.diegotorresd.SonoPhone`.

La UPM está enrolada en un University Program, gestionado desde el Departamento de Arquitecturas Telemáticas (DIT) de la ETSIT. El profesor Santiago Pavón cuenta en su web con un completo curso sobre iOS que comienza por detallar el proceso de solicitar la inclusión del desarrollador en el equipo y la generación de un perfil de aprovisionamiento para su terminal (Gómez, 2012, ver).

Entrando en contacto con él fue posible dar de alta un iPhone para probar SonoPhone mediante la obtención del perfil de aprovisionamiento adecuado.

Apéndice E

Almacenamiento en la nube con Amazon S3

E.1. Introducción al servicio

Amazon S3 es una parte de Amazon Web Services (AWS), una de las infraestructuras de *cloud computing* más importantes actualmente. S3, que significa *Simple Storage Service* (servicio de almacenamiento simple), es una plataforma de almacenamiento que permite subir datos y acceder a ellos de manera sencilla, eficiente y fiable.

Numerosas aplicaciones y sitios web utilizan S3, como por ejemplo Dropbox, Tumblr o Foursquare. El lector interesado en conocer en detalle el funcionamiento de S3 puede consultar la *Developer Guide* (ver Bibliografía).

E.2. Darse de alta

Para utilizar cualquier servicio de AWS, es necesario en primer lugar registrarse en Amazon. Este registro requiere introducir datos de pago, aunque no se factura hasta que se utilizan los servicios. Se ofrece una capa gratuita (*free tier*) que incluye, entre otros muchos servicios, 5 GB de espacio en S3[1].

[1] La lista completa se puede consultar en `http://aws.amazon.com/free/` (última consulta, Junio 2013)

E.3. Funcionamiento

S3 se estructura en depósitos (*buckets*) que contienen objetos. Un usuario puede crear, destruir y modificar sus depósitos, y efectuar estas mismas operaciones con los objetos que contienen. Cada objeto está definido por su ruta de acceso, y además pueden especificarse permisos, como se describe en la sección E.3.2.

Un depósito puede encontrarse en cualquiera de las regiones que define Amazon, repartidas por el mundo, para optimizar la latencia de acceso o por razones legales o de otra índole. Sin embargo, este detalle es prácticamente transparente a la API.

E.3.1. API RESTful

Para interactuar con S3, se envían peticiones HTTP al servidor, indicando la ruta sobre la que se quiere actuar y la acción que se desea hacer.

La API de S3 sigue la arquitectura REST. REST (siglas de *REpresentational State Transfer*) es un paradigma acuñado por Roy Fielding para hacer las aplicaciones web más simples y escalables, intentando unificar la manera en la que se implementan los servicios web aprovechando las capacidades del protocolo HTTP.

Algunos de los principios básicos de REST son:

Todo recurso tiene un identificador (URI) asociado En vez de mantener privados los identificadores de los recursos que usa la aplicación, en REST se da acceso mediante URIs a éstos. Por ejemplo, en una aplicación, `/product/20342935` es una URI que identifica a un producto concreto. En S3, la URI de un objeto está formada por el nombre del depósito y el del objeto.

Verbos HTTP En vez de implementar en cada aplicación una arquitectura para realizar las operaciones básicas o CRUD[2], REST propone que se usen las acciones o *verbos* que HTTP ya implementa: principalmente GET, POST, PUT, y DELETE. GET debe ser una acción "segura" e idempotente, es decir, no producir ningún efecto en el estado de los datos, mientras que los otros verbos realizan una modificación.

No se mantiene el estado El servidor no debe mantener ningún estado para "recordar" al cliente entre peticiones: todo el estado debe mantenerse en éste y adjuntarse a la petición.

[2]Siglas de *Create Read Update Delete*.

Hipermedia Las respuestas del servidor pueden contener enlaces que referencien a otras URIs ya sean dentro de la misma aplicación o fuera. De este modo se aprovecha para la comunicación cliente-servidor la fuerza principal de HTTP, el hipermedia.

Una API que se adhiere a los principios de REST se denomina RESTful. En E.1 se muestra un ejemplo de petición HTTP para borrar un objeto en S3, concretamente la imagen puppy.jpg del depósito mybucket. La cabecera "Authorization" define la autenticación, que se explica en la sección E.3.2.

Listado E.1: Petición HTTP de ejemplo

```
DELETE /puppy.jpg HTTP/1.1
User-Agent: dotnet
Host: mybucket.s3.amazonaws.com
Date: Tue, 15 Jan 2008 21:20:27 +0000
x-amz-date: Tue, 15 Jan 2008 21:20:27 +0000
Authorization:
     AWS AKIAIOSFODNN7EXAMPLE:k3nL7gH3+PadhTEVn5EXAMPLE
```

E.3.2. Autenticación

Una de las partes más sutiles de S3 es su mecanismo de autenticación, para asegurar que solo accede a los depósitos quien el desarrollador pretende. Toda la autenticación se basa en un sistema de claves asimétricas, en el que el cliente encripta el mensaje con su clave privada, S3 lo desencripta con la pública y comprueba que es correcto.

Existen tres niveles de autenticación:

- Cada usuario tiene una clave privada con privilegios de administrador sobre todos los depósitos que crea. Sin embargo, no es recomendable su uso cuando otros usuarios deban acceder a los depósitos del usuario, ya que implicaría darles todo el control.
- Se proporciona un mecanismo denominado IAM (*Identity and Access Management*) para crear usuarios con permisos determinados. Sin embargo, este proceso es imprácticto cuando el desarrollador no puede controlar qué usuarios van a acceder a sus depósitos.
- El modo recomendado de proporcionar a los usuarios de una aplicación acceso a unos depósitos S3 propiedad del desarrollador es la utilización de una "máquina

Listado E.2: Código Objective C para añadir un objeto a un depósito

```
NSData *data = [objectData.text
        dataUsingEncoding:NSUTF8StringEncoding];
S3PutObjectRequest *request = [[[S3PutObjectRequest alloc]
        initWithKey:objectName.text
        inBucket:self.bucket] autorelease];
request.data = data;
S3PutObjectResponse *response = [[AmazonClientManager s3]
         putObject:request];
```

expendedora de tokens" (*Token Vending Machine*, TVM). Son aplicaciones web que proporcionan una clave de acceso (token) al usuario que lo solicite. Existen dos tipos: la TVM con identificación, que requiere que el usuario se registre, o la TVM anónima, que usa la UUID del dispositivo como identificación[3]. Amazon provee dos TVMs en forma de aplicaciones web Java que se pueden desplegar en un servidor propio o bien utilizando un servicio de AWS llamado Elastic BeanStalk. De esta forma, sólo hay que proporcionar a la aplicación la URL donde está alojada la TVM, y al realizar las peticiones se pedirá el token necesario.

E.4. El SDK para iOS

Amazon proporciona un SDK para iOS que consta de varios Frameworks y código de ejemplo, que hace muy fácil la implementación de almacenamiento en nube en las aplicaciones móviles.

Por ejemplo, el código para añadir un objeto a un depósito se reduce al mostrado en E.2. En la figura E.1 se puede ver la aplicación de prueba de AWS.

[3]En SonoPhone se ha usado esta última.

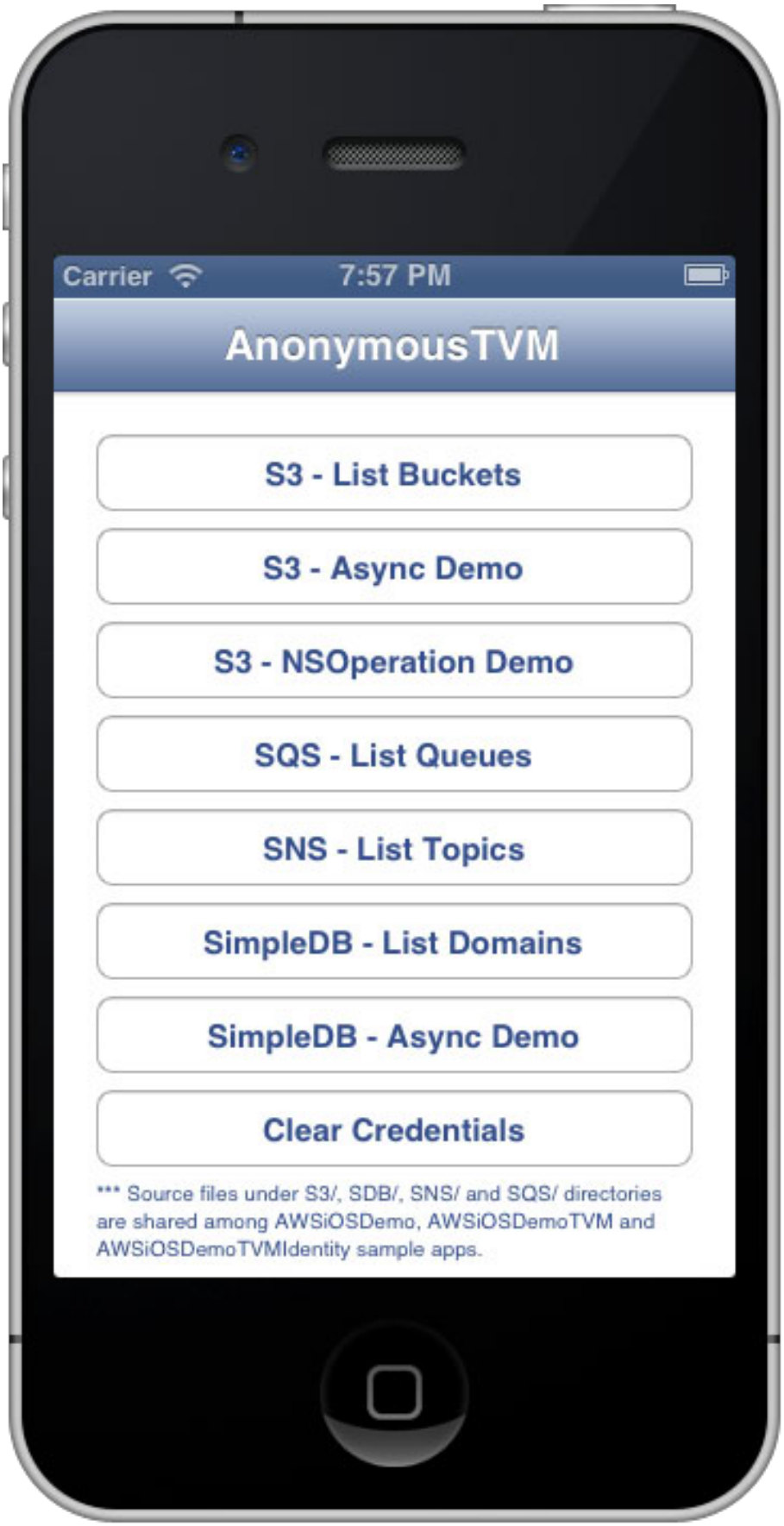

Figura E.1: Aplicación de ejemplo de AWS con TVM anónima, corriendo en el simulador de iPhone

Bibliografía

Apple Inc. iOS App Programming Guide. http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/, 2013a.

Apple Inc. iOS Human Interface Guidelines. http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/, 2013b.

Brixen, E. *Audio Metering: Measurements, Standards and Practice*. Focal Press, 2012. ISBN 978-0-240-81467-4.

Centro Español de Metrología (CEM). Guía para la expresión de la incertidumbre de medida. Informe técnico, CSIC, 2000.

Fahy, F. *Foundations of engineering acoustics*. Academic Press, 2003.

Gartner. "Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013". 2013.

Gómez, S. P. Desarrollo de apps para iOS: Introducción. http://web.dit.upm.es/~santiago/docencia/ios/011-Introduccion-20121025.pdf, 2012.

IDC. "IDC Forecasts Worldwide Tablet Shipments to Surpass Portable PC Shipments in 2013, Total PC Shipments in 2015". 2013.

International Communications Union. "ITU Internet report: the Internet of Things". Informe técnico, ITU, 2005.

Kuehn, J. *Instrumentation Reference Book*, capítulo Sound Measurement. 0-7506-8308-2. Butterworth-Heinemann, 4th edición, 2009.

Thomas, M. MLS theory. http://www.commsp.ee.ic.ac.uk/~mrt102/projects/mls/MLS%20Theory.pdf, 2009.

UNE-EN. Electroacústica. Sonómetros. Informe Técnico 61672-1, AENOR, 2005.